中国科技期刊卓越行动计划项目入选期刊

控制与决策

CONTROL AND DECISION

边缘计算环境中基于规划图的服务组合修复方法

高志浩, 李静, 祝铭, 刘彤阳, 鲁瑞

引用本文:
高志浩,李静,祝铭,刘彤阳,鲁瑞. 边缘计算环境中基于规划图的服务组合修复方法[J]. 控制与决策, 2024, 39(7): 2438−2446.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2023.1599

边缘计算环境中基于规划图的服务组合修复方法

高志浩[1], 李 静[1], 祝 铭[1†], 刘彤阳[1], 鲁 瑞[2]

(1. 山东理工大学 计算机科学与技术学院，山东 淄博 255000；
2. 澳大利亚国立大学 工程与计算机科学学院，堪培拉 2601)

摘 要: 在云计算和边缘计算环境中,通过提取和组合可用的服务以满足用户需求已成为常见的做法. 然而,当前的方法难以应对由于用户需求变化或外部环境变动导致的组合失效问题. 为了应对这一挑战,提出一种边缘计算环境中基于规划图的服务组合及其修复方法. 首先,结合移动路径模型和规划图方法完成服务组合过程,通过规划图的构建可以有效地评估和选择适合用户需求的服务组合. 当服务集合发生变化或用户目标更改时,能够在现有的规划图基础上生成新的解决方案,以满足用户的需求. 这种修复方法能够实时适应云边环境中的变化,提高系统的灵活性和可靠性. 经过实验测试发现,所提出的修复方法相较于重新规划具有更好的性能表现,并验证了修复方法在组合失效问题上的有效性和实用性.

关键词: 服务组合；规划图；服务组合修复；边缘计算；服务质量；服务时延

中图分类号: TP338.8 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2023.1599

引用格式: 高志浩,李静,祝铭,等. 边缘计算环境中基于规划图的服务组合修复方法[J]. 控制与决策, 2024, 39(7): 2438-2446.

A repairing service composition method based on planning graph under edge computing

GAO Zhi-hao[1], *LI Jing*[1], *ZHU Ming*[1†], *LIU Tong-yang*[1], *LU Rui*[2]

(1. School of Computer Science and Technology，Shandong University of Science and Technology，Zibo 255000，China；
2. School of Engineering and Computer Science，Australian National University，Canberra 2601，Australia)

Abstract: In cloud and edge computing environments, it is common to extract and combine available services to meet user needs. However, current methods struggle to cope with composition failures caused by changes in user needs or external environment. To address this challenge, this paper proposes a planning-based service composition and repair method in edge computing environments. We first combine the mobile path model and planning graph method to complete the service composition process. The construction of the graph allows for efficient evaluation and selection of service compositions that suit user demands. When the service set changes or user goals are modified, the method can generate new solutions based on the existing planning graph to meet user needs. This repair method can adapt to changes in real-time in the cloud-edge environment, enhancing system flexibility and reliability. Experimental tests have shown that the proposed repair method outperforms replanning, demonstrating its effectiveness and practicality in addressing combination failures.

Keywords: service composition；planning graph；repair service composition；edge computing；Qos；service delay

0 引 言

近年来,随着科技的发展,越来越多的人开始使用智能设备[1]. 与此同时,许多对时延要求较高以及需求密集型应用的数量也在迅速增加. 传统的云计算在面对数据爆炸和更高的服务质量要求方面已经不能完全满足用户的需求. 此外,云计算中心化架构也存在单点故障风险,并且难以满足对敏感性和低延迟的需求[2].

为了应对这些挑战,移动边缘计算技术(mobile edge computing, MEC)被提出[3-4],其采用分布式架构,将计算资源分散到多个边缘设备上,并且可以将计算、存储和网络资源放置在离用户更近的位置,使得

收稿日期: 2023-11-16；**录用日期:** 2024-01-02.

基金项目: 国家自然科学基金项目(62002205)；嵌入式与网络计算湖南省重点实验室开放基金项目(20220106)；教育部高等学校科学研究发展中心中国高校产学研创新基金——新一代信息技术创新项目(2022IT048).

责任编委: 邓庆绪.

†通讯作者. E-mail: zhu_ming@sdut.edu.cn.

用户可以在较短的响应时间内访问到各种所需的服务. 其中,用户移动过程会提交服务请求,数据、算力、资源等都可被包成服务,供用户按需请求. 在云边环境中,由于用户需求的复杂性,单个服务往往无法满足要求,需要通过整合和共享多个服务资源进行服务组合. 然而,服务间关系复杂且用户需求各异,这给服务组合和规划带来了很大的挑战. 为应对这个问题,AI规划方法[5-8]应运而生. Li等[9]提出了一种关系数据库方法,用于自动化服务组合. 该方法预先生成和存储所有可能的服务组合,并在用户请求时通过查询数据库返回最优的服务. Zhu等[10]提出了一种综合服务质量(quality of service, QoS)和修改规划图的Web服务组合方法,以应对复杂性和耗时性. 通过模糊逻辑处理不确定性并支持决策,该方法在服务修剪和解生成方面进行优化,取得了更好的解决方案. 韩敏等[11]提出一种改进的多目标灰狼优化(improved multi-objective grey wolf optimization, IMOGWO)算法来求解Web环境下的服务组合问题. 然而,这些方法只能提供一次性的解决方案,无法适应服务或用户需求的变化.

为了应对动态变化的服务和用户需求的挑战,研究者们提出了多种方法. 其中: Friedrich等[12]提出了一种自愈方法处理服务流程中的异常,并使用基于模型的方法修复故障活动; Church等[13]研究了服务组合中的故障问题,并提出了一个形式化的模型和相应的修复建议,设计了一种支持程序员调试服务故障的技术; Eder等[14]提出了一种架构,将活动和实例视为可管理的资源,用于修复故障流程. 然而,这些方法无法适用于移动边缘环境. 因此,本文使用规划图方法描述云边环境下的服务组合问题,并引入修复算法减少用户等待时间和延迟.

规划图是一种图形化工具,能够很好地建模服务组合问题,其能够有效表示服务之间的依赖关系和约束条件,并描述服务的属性和行为. 这种方法的有效性已在Web服务环境中得到验证[15]. 然而,他们的方法具有局限性,没有进一步考虑服务QoS对整个维修过程的影响. 本文将其改进并应用在云边环境中来描述边缘环境中的服务组合. 当服务不可用、损坏或用户需求发生变化时,规划图的一部分仍然有效,因此本文通过修复原有规划图来解决这一问题,而不是只提供一次性的解决方案,避免了重新规划(replanning). 规划图作为一种灵活且高效的工具,能够应对云边服务环境中的不确定性和变化. 所以,使用规划图可以有效解决云边服务中的服务组合和规划问题并且应对不断变化的服务环境.

这种基于规划图的修复方法能够提供灵活且高效的服务组合解决方案,适应云边服务环境中的动态变化和不确定性. 本文主要工作总结如下:

1) 本文考虑了在云和边缘计算中的移动服务组合问题,并使用规划图来描述这一问题,其可以描述服务之间的关系,并且这种应用规划图的方法在云边环境中还没有被探讨过.

2) 本文提出了一种基于规划图的服务组合修复方法. 在移动边缘计算环境中,服务可能会因网络变化、资源限制或其他因素而发生故障或中断. 本文的修复方法能够动态地调整服务组合,以适应变化的环境,并尽快恢复系统的正常运行.

1　背景知识

1.1　规划图

规划图PG是一个有向无环的水平图[16],交替包含问题层 P_i 和动作层 A_i. 其中: A_i 层包含动作 a, P_i 层中包含 a 的前提条件 $\text{preconds}(a)$ 与 a 的后续结果 $\text{effects}(a)$. 通过组合问题层和动作层,可以形成规划图的结构.

首先进行前向扩展,规划图逐渐从初始状态 S_0 向目标状态goal拓展. 在前向扩展的过程中,问题层和动作层不断扩展,建立问题状态和动作之间的联系. 当规划图满足终止条件时,即问题层中包含全部目标集合或最后两层问题层的命题层相同,前向扩展过程停止. 规划图构建完成后,使用反向搜索来寻找满足目标的服务组合解决方案. 反向搜索从目标状态开始,逆向回溯到初始状态并跟踪每个状态的动作,最终得到一个满足目标的组合方案.

1.2　服务定义

本文将服务 w 定义为一个4元组 $(p, w_{\text{in}}, w_{\text{out}}, q)$. 其中: p 表示服务 w 所部署的服务器位置,它可以位于云服务器或者位于边缘服务器; w_{in} 表示服务 w 所需要的输入参数; w_{out} 表示服务 w 的输出参数; q 表示服务 w 的服务质量(QoS). 本文中使用响应时间 (T) 作为衡量服务QoS的标准.

假设一个人脸验证服务位于云服务器 (p),响应时间为12 ms (q),用户需要提供面部视频信息 (w_{in}),之后服务返回用户所需的信息 (w_{out}). 此外,本文定义这些输入输出信息的数据大小为Datasize,这些输入输出信息的集合称为Concept,大小单位为Mb.

1.3　服务组合的定义

在云边环境中为了能更好地满足用户需求,往往需要将多个服务的资源进行共享和利用,即进行服务组合. 本文定义一个服务组合问题如下:

1) W 是一组有限的服务集合;

2) C_{in} 是输入参数的有限集合;

3 C_{out} 是输出参数的有限集合;

4) Q 是服务质量标准集合.

服务组合问题可以映射到规划图中,表示如下:

$$w \leftrightarrow a,\ w_{\text{in}} \leftrightarrow \text{preconed}(a),\ w_{\text{out}} \leftrightarrow \text{effects}(a),$$

$$C_{\text{in}} \leftrightarrow S_0,\ C_{\text{out}} \leftrightarrow \text{goal}.$$

给出两个服务 w_i, w_j,它们之间有3种服务执行结构[10]: $w_i; w_j$ 表示顺序执行结构, $w_i \| w_j$ 表示并行执行结构, $w_i | w_j$ 表示选择执行结构.

1.4 移动模型

随着边缘计算服务的普及程度不断提高,越来越多的服务请求发生在边缘环境中. 而用户的移动路径对服务上传、选择以及服务组合结果的得到时间起着重要影响,最终也会影响到服务组合解决方案的质量,因此需要对移动路径进行建模.

假设用户初始距离基站的距离为 D,移动的方向角为 θ,且用户以一个恒定的速度移动,在经过时间 t 之后到达最终位置,此时用户和基站的距离[17]为

$$d = \sqrt{(vt)^2 - 2Dvt\cos\theta + D^2}, \tag{1}$$

其中 v 为用户的移动速度. 本文将用户初始位置和移动速度相结合,并将用户在时间 t 内的移动路径考虑进去,从而得到用户与基站之间的实际距离.

通过对移动路径的建模,本文可以更加准确地预测用户与基站之间的距离,从而为服务上传、选择和服务组合的过程提供更合理的参考. 这将有助于改善服务组合解决方案的质量,并提高用户移动环境下的服务体验.

2 云边环境下服务组合

本节主要介绍云边环境中基于规划图的服务组合过程,包括从用户上传到下载结果的计算方法.

图1中展示了本文服务组合方法的主要结构,具体内容如下.

移动边缘和云环境: 主要模拟了云服务器以及边缘服务器的分布,并且服务分布到各个服务器中. 每个边缘服务器都连接到互联网,并能够与其他服务器进行数据通信. 同样,每个云服务器都连接到互联网,并能够与其他服务器传输数据. 在移动边缘和云计算环境下,用户的移动路径部分决定了上传、服务组合和下载的时间. 因此,本文对用户的移动路径进行建模.

基于规划图的服务组合: 移动用户将他的请求以及输入上传到附近的基站,接着将其上传到服务器; 之后在云边服务环境中进行规划图的前向拓展,直到找到满足用户需求的输出结果; 最后用户从靠近他的基站接收解决方案.

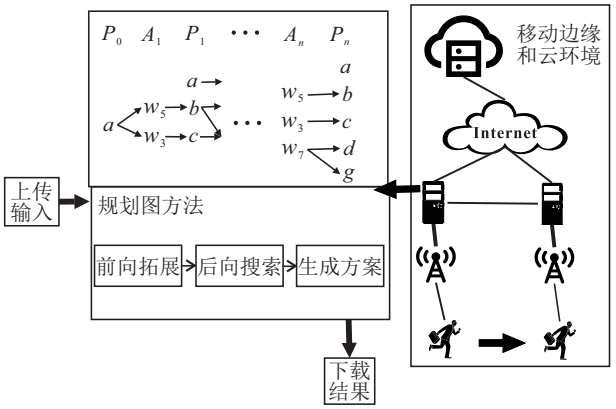

图1 云边环境下基于规划图的服务组合

在云边环境中,服务器之间的异构性是存在的. 当用户使用这些服务时,由于服务的差异,它们在服务组合中所需的资源和时间也不同. 本文定义在云边环境中总共的响应时间为 T_{all}, 是用户从发出请求到接收到解决方案的总时间,可由下式计算:

$$T_{\text{all}} = T_{\text{upload}} + T_{\text{comp}} + T_{\text{download}}. \tag{2}$$

其中: T_{upload} 是指上传请求的时间, T_{download} 是下载解决方案的时间, T_{comp} 是服务组合的响应时间. 对于组合时间本文将在案例说明中介绍具体的计算过程.

对于上传请求的时间,包括用户请求传输到基站的时间 T_{u2b} 以及基站到边缘服务器的传输时间 T_{b2e}. T_{upload} 表示为

$$T_{\text{upload}} = T_{u2b} + T_{b2e}. \tag{3}$$

对于下载时间,如果最后一个任务的输出结果不在用户所处范围内的服务器中,可能在边缘或者云服务器中,则下载时间包括服务器之间的传输时间 T_{s2s},服务器到基站的传输时间 T_{s2b},基站到用户的传输时间 T_{b2u}. T_{download} 表示为

$$T_{\text{download}} = T_{s2s} + T_{s2b} + T_{b2u}. \tag{4}$$

请求上传到基站 T_{u2b} 的时间和从基站 T_{b2u} 下载结果的时间计算如下:

$$T_{u2b} = \sum \text{Input}_{\text{Datasize}}/r(B,g), \tag{5}$$

$$T_{b2u} = \sum \text{Goal}_{\text{Datasize}}/r(B,g). \tag{6}$$

其中: $\sum \text{Input}_{\text{Datasize}}$ 表示上传的任务的大小, $\sum \text{Goal}_{\text{Datasize}}$ 表示组合目标结果的大小; $r(B,g)$ 表示可实现的数据传输速率,依据香农公式,其可由下式计算得到:

$$r(B,g) = B\log_2(1 + \text{tp}g/\sigma^2). \tag{7}$$

σ^2为接收机处的噪声功率; tp为设备的无线发射功率; g为信道功率增益,可计算为$h_0 d^{-\gamma}$, h_0为接收功率,d为设备与服务器之间的距离,γ为路径损耗系数.

服务器之间的传输时间T_{tran}也取决于数据大小和带宽BW_{s2s},可由下式计算:

$$T_{\text{tran}} = \text{Concept}_{\text{Datasize}}/\text{BW}_{s2s} \tag{8}$$

3　修复模型

3.1　服务修复定义

在现实世界中,由于用户需求的改变或用户移动而带来的变化都可能导致可用服务和服务组合发生变化. 为了解决这个问题,文献[15]提出了一种在Web服务环境中的修复方法. 他们通过一种基于人工智能的方法来修复受损的规划图,并获得更新的服务组合方案. 然而,他们的方法具有局限性. 他们认为损坏或消失的服务产生的所有参数都必须修复. 此外,该方法没有进一步考虑服务QoS对整个修复过程的影响.

本文对该方法进行了改进与调整,提出了一种规划图修复方法(repairing a damaged planning graph for service composition, RDPGSC),使其可以应用到边缘计算环境中. 本文的方法仍然基于原始计划,因为本文相信尽管规划图已经被破坏,但仍然有很多计划是可行的,即大部分的服务还可以被再次使用,这可以为后续的工作提供帮助.

为了方便描述修复过程,这里给出以下定义: G表示被破坏后的规划图, BP表示在规划图中一些服务未满足的输入, g表示goal集合中需要被修复的元素, W表示候选服务集合.

本文主要通过在更新后的候选服务集合W中选择w修复规划图,根据下式选择服务:

$$\begin{aligned} f_1(G,w) = {} & |g \bigcap w_{\text{out}}| + \alpha |P_m \bigcap \text{BP} \bigcap w_{\text{out}}| + \\ & |P_{m-1} \bigcap w_{\text{in}}| - |w_{\text{in}} - P_{m-1}| - \\ & \frac{(T_w - T_W^{\min})\beta}{T_W^{\max} - T_W^{\min}} - w_{\text{outsize}}\delta. \end{aligned} \tag{9}$$

式(9)用来计算规划图中除了第A_n层即最后一层(n层)服务的适应度值. $P_m(0 \leqslant m < n)$表示正在修复的当前层; $|P_m \bigcap \text{BP} \bigcap w_{\text{out}}|$表示$w$的输出的元素可以满足在第$P_m$层的BP集合里所需要参数的数量; $|P_{m-1} \bigcap w_{\text{in}}|$表示的是在第$P_{m-1}$层中元素中不能满足$w$输入的数量; $|w_{\text{in}} - P_{m-1}|$表示$P_{m-1}$层中不能满足$w$输入的参数的个数; T_w表示服务的响应时间; $T_W^{\max}$和$T_W^{\min}$表示所有服务的最大响应时间和最小响应时间,这一部分表示对候选服务响应时间的归一化计算; w_{outsize}表示服务需要传输到其他服务器的最大输出数据的大小; α, β和γ分别为权重系数.

$$\begin{aligned} & f_2(G,w) = \\ & |g \bigcap w_{\text{out}}|\alpha + |P_{n-1} \bigcap w_{\text{in}}| - |w_{\text{in}} - P_{n-1}| - \\ & \frac{(T_w - T_W^{\min})\beta}{T_W^{\max} - T_W^{\min}} + \text{isNear}. \end{aligned} \tag{10}$$

式(10)是被用来计算第A_n层即最后一层服务的适应度值. 其中: $|g \bigcap w_{\text{out}}|$表示候选服务$w$可以满足的目标集(goal)得分; $|P_{n-1} \bigcap w_{\text{in}}|$表示在第$P_{n-1}$层的元素中不能满足$w$输入的数量; isNear用于判断最后一层服务层中所选的服务是否位于用户当前基站范围内的服务器中,若位于范围内,则可以缩短数据传输时间,其数值为0或1.

3.2　案例说明

本节将使用一个简单的例子来展示在云边环境中使用规划图进行服务选择的过程以及服务修复的具体过程.

首先介绍规划图的前向拓展过程. 图2的例子中为代驾司机接单的过程,具体为司机接收订单后上传自己的信息集合S_0,获取从他的位置到客户位置以及到达客户的目的地的路径信息. 表1～表3分别为事例中的参数信息以及服务描述,包括服务的输入和输出以及这些数据的大小(单位为Mb). 首先代驾司机会将初始输入上传到最近的基站中. 开始时,代驾司机距离基站最近为100 m,其中代驾司机的初始输入集合S_0里包括Routetodes和RoutetoC,它们上传到基站的时间可以计算为

$$T_{u2b} = \sum \text{Input}_{\text{Datasize}}/r(B,g) = 853\,\text{ms}.$$

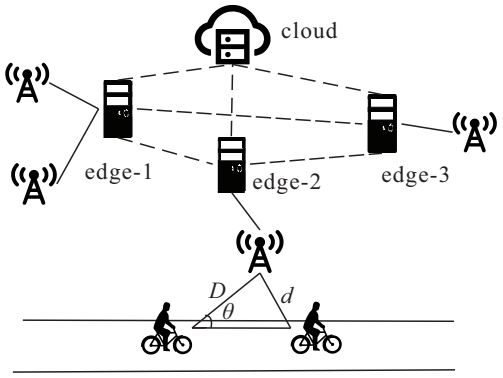

图2　案例图示

表1　参数描述

参数名	参数值	参数名	参数值
σ^2	−60 dBm/Hz	γ	2
B	256 MHz	tp	500 MW
h_0	−30dB	BW_{c2c}	5 Gbps
BW_{e2e}	3 Gbps	BW_{e2c}	1.5 Gbps
BW_{e2b}	2 Gbps	θ	30°

表2 案例数据大小

Concept	Datasize	Concept	Datasize
Dinfo	2	Routetodes	5
Dcredit	0.1	GPSinfo	5
DID	0.1	Cloact	2
Dsite	5	Cdest	1
Faceinfo	5	Drivepath	5
Cphone	5	Csite	1
Cdestion	1	RoutetoC	5
Cinfo	3		

边缘服务器到基站之间的传输速度BW_{b2e}为2 Gbps. 因此,输入上传到边缘服务器的时间可以计算为

$$T_{b2e_2}=\sum \mathrm{Input}_{\mathrm{Datasize}}/\mathrm{BW}_{b2e}=3.5\ \mathrm{ms}.$$

得到T_{upload}的时间为

$$T_{\mathrm{upload}}=T_{u2b}+T_{b2e}=856.5\ \mathrm{ms}.$$

然后,计算在云边环境中基于规划图的服务组合的响应时间T_{comp}.此时初始输入S_0在Edge-2中,接着服务exDinfo调用输入Dinfo,而服务exDinfo位于云服务器,因此需要计算出Dinfo的传输时间T_{tran}为1.3 ms,此外exDinfo的响应时间T为12 ms,通过计算可得Dcredit, DID产生的时间为13.3 ms.

表3 案例服务描述

服务	w_{in}	w_{out}	位置	响应时间/ms	描述
extDinfo	Dinfo	Dcredit,DID	Cloud	12	获取司机信息
Locating1	Dinfo	Dsite	Edge-1	28	获取司机位置
FaceVer	Dinfo,Facepic	Cinfo	Edge-2	27	人脸验证
Locating2	Cinfo	Csite	Edge-2	20	获取客户位置
extPhone	Cinfo	Cphone	Edge-3	10	获取客户手机信息
extDest	Cinfo	Cdestion	Edge-2	12	获取客户目的地
getGPS	Cphone	CGPSinfo	Edge-3	24	获得GPS信息
findPathtoC	Dsite,Csite	RoutetoC	Edge-3	15	获取到达用户路径
findPathtodes1	Cdestion,Csite	Routetodes	Edge-2	21	获取用户所在位置到用户目的地路径
findPathtodes2	Clocat,Cdest	Routetodes	Edge-2	19	获取用户所在位置到用户目的地路径
findPathtodes3	Clocat,Cdest	Routetodes	Cloud	24	获取用户所在位置到用户目的地路径
extLocat	GPSinfo	Clocat	Edge-2	16	获取客户位置

此外,规定同一个服务器中的传输时间为0 ms. 计算可得Cinfo的产生时间为27 ms. 经过上述计算可得Dsite的产生时间为29 ms. 至此,P_1中产生Concept的时间就计算完毕,接着继续进行下面的规划图拓展过程,P_2层中参数产生时间的计算与前一层大致相同,这里直接给出结果,得到Csite, Cphone和Cdestion的时间为47 ms, 38 ms和39 ms.

最后,计算最后一层中得到RoutetoC的时间,其需要的输入为Dsite以及Csite,因此在调用服务之前需要它们全部传输到服务器Edge-3中,此时还需要计算两者到达时间29 ms+1.67 ms < 47 ms+0.33 ms,因此得到RoutetoC的时间为47 ms + 0.33 ms + 15 ms = 63.3 ms. 同时计算得到Routetodes的时间,结果为68 ms. 对比之后,最晚得到的结果是Routetodes,因此服务组合的响应时间总共为68 ms. 最后用户需要从服务器Edge-3和Edge-2中下载结果集合.

假设用户速度为10 m/s,此时根据式(1)计算,用户距离基站的距离d为92.1 m. 因此,需要将RoutetoC从Edge-3传到Edge-2中,时间为

$$T_{s2s}=\mathrm{RoutetoC}_{\mathrm{Datasize}}/\mathrm{BW}_{e2e}=1.67\ \mathrm{ms}.$$

接着将结果集合传入基站,有

$$T_{e2b}=\sum \mathrm{Goal}_{\mathrm{Datasize}}/\mathrm{BW}_{e2b}=1.67\ \mathrm{ms}.$$

用户从基站中下载结果的时间为

$$T_{b2u}=\sum \mathrm{Goal}_{\mathrm{Datasize}}/r(B,g)=1\,206.4\ \mathrm{ms}.$$

总共的下载时间为

$$T_{\mathrm{download}}=T_{s2s}+T_{e2b}+T_{b2u}=1\,209.7\ \mathrm{ms}.$$

用户从输入需求到得到结果总共需要的时间为

$$T_{\mathrm{all}}=T_{\mathrm{upload}}+T_{\mathrm{comp}}+T_{\mathrm{download}}=2\,134.2\ \mathrm{ms}.$$

图3为案例的规划图前向拓展之后的结果. 这里直接给出规划图反向搜索后的服务组合结果为:{(Locating1||FaceVer); (Locating2||exDest); (findPathtoC||findPathtodes1)}.

因为一些原因,当服务findPathtodes1被破坏之后,使得原始的规划图不可再被使用,即不能再提供

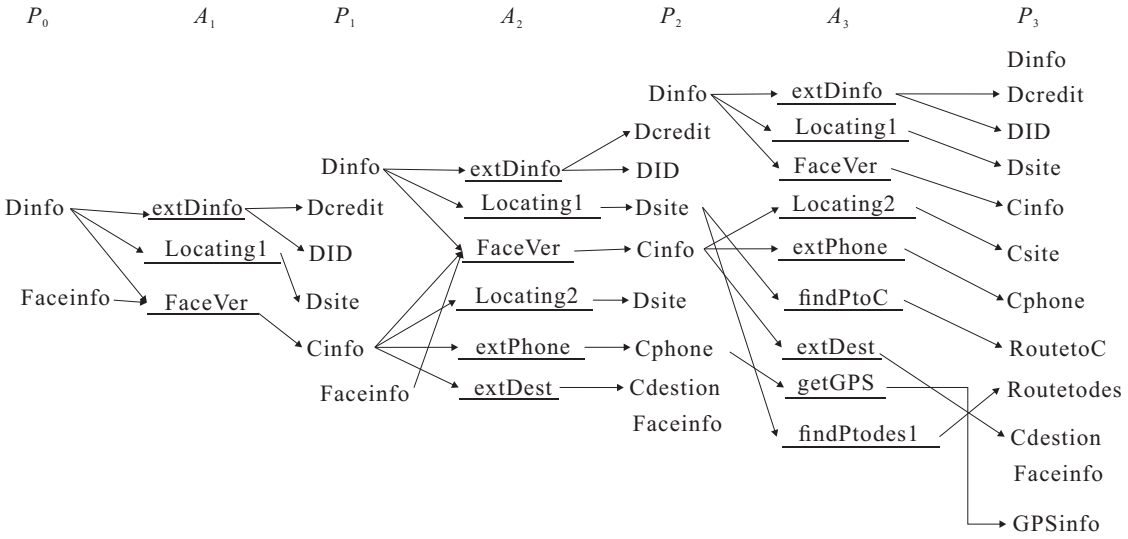

图 3　案例规划图图示

之前的组合方案,此时将会在候选服务集合中选取合适的服务来修复破损的规划图.

$$f_2(G,\text{findPathtodes3})=10+1-1-0.67+0=9.33,$$

$$f_2(G,\text{findPathtodes2})=10+1-1-0.5+1=10.5.$$

此时使用式(10)进行计算. 在最后一层中,两个服务都能够生成并且仅生成Routetodes. 基于权重因素计算findPathtodes2和findPathtodes3,$|g\bigcap w_{\text{out}}|\alpha$都为10. 在$P_{n-1}$层中,能够满足findPathtodes2和findPathtodes3的输入为Csite,因此$|P_{n-1}\bigcap w_{\text{in}}|$的结果为1. 两个服务都有一个输入无法被满足,因此$|w_{\text{in}}-P_{n-1}|$的计算值为1. 接着,对响应时间进行归一化计算(参见表3),分别为0.67和0.5,用户目前距离最近的基站处于Edge-2的范围内,相较于在Cloud服务器中的findPathtodes3,处于Edge-2中的Routetodes2距离用户更近,因此isNear计算为1.

将适应度更高的findPathtodes2加入A_5,并将其所需要的输入Cloact加入到P_4层中,此时g集合为空,BP集合中加入Cloact. 接着,以相同的方法将getGPS加入A_3层中,此时BP集合中仍有元素Cphone,不为空,这时就需要在规划图前面加入空的P层以及A层,通过添加服务extPhone,修复BP集合中的元素,添加FaceVer修复Cinfo,BP为空,修复过程结束. 图4为修复完成之后的规划图.

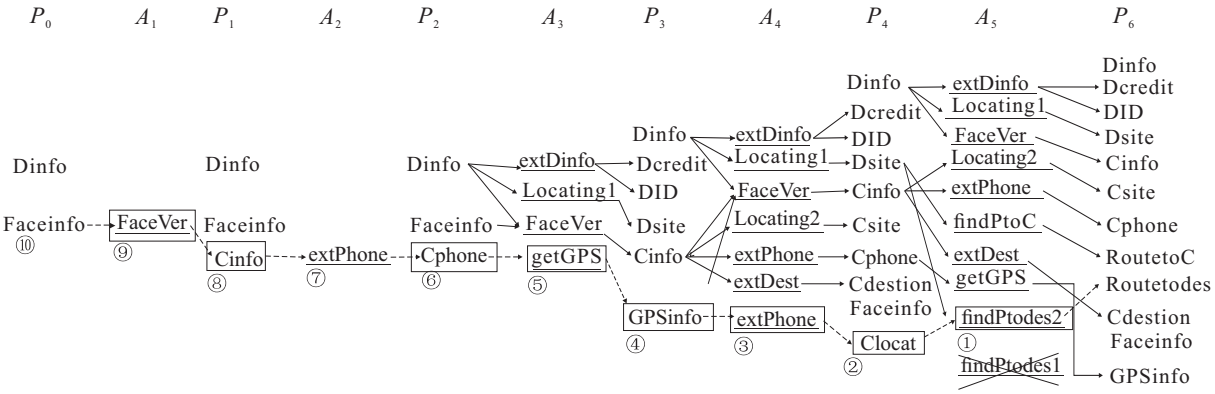

图 4　规划图修复示例

3.3　服务修复算法介绍

本文通过计算适应度值不断修复规划图,直到最终为用户返回服务结果,算法流程见算法1,修复规划图的时间开销为$O(n|W|)$,最终将其加入用户得到结果的总时间进行计算,因此用户得到结果的时间不仅包括上传、下载以及服务组合的时间,还包括修复规划图的时间开销.

首先,从最后一层P_n开始,按照式(10)选择合适的服务w,先修复与目标g相关的部分,使服务能够生成g需要的输入前置参数. 一旦找到合适的候选服务w,则将其添加到层A_n,并将w所需的输入(w_{in})添加到层P_{n-1}中. 然后,更新BP(第1行～第9行). 接下来,按照式(8)修复从层P_{n-1}到层P_1即P_m的规划图,修复和更新BP中的元素(第10行～第28行). 修复过

程与前一部分大体相同,但是在这一部分本文不会因为某些不可被修复的元素而被迫停止修复过程,而是将其暂时存储到BPQ中,并在规划图中删除被其影响的服务以减少后续后向搜素空间以及返回结果的时间(第22行～第27行). 当第一个P层被修复,但BP仍然不为空时,算法继续创建一个空的P层和一个空的A层,并按照式(9)修复这些层,修复过程也与前面类似,直到BP为空. 其中BPH用来记录每层中的未被满足的参数,当BP与BPH有重合时,规划图修复失败,若成功则返回一个服务组合结果(第29行～第45行).

算法1 修复损坏后的规划图并返回一个解决方案.

输入: 待修复的规划图G,规划图G中一组不满足的服务的输入集合BP,一组无法修复的参数集合BPQ,一组候选服务W,一组未满足输入的goal集合g;

输出: 生成一个修复完成的规划图或者返回失败.

step 1: while $g \neq \emptyset$ do
step 2: 根据式(10)在W中选择w
step 3: if $w \notin W$ then break
step 4: $A_n \leftarrow A_n \bigcup w$
step 5: $P_n \leftarrow P_n \bigcup w_{\text{out}}$
step 6: $P_{n-1} \leftarrow P_{n-1} \bigcup w_{\text{in}}$
step 7: $g \leftarrow g/w_{\text{out}}$
step 8: BP $\leftarrow$ BP$\bigcup(w_{\text{in}}/P_{n-1})$
step 9: end while
step 10: If $g \neq \emptyset$ then return fail
step 11: for each P_m in G
step 12: while BP$\bigcap P_m \neq \emptyset$ do
step 13: 根据式(9) 在W 中选择w
step 14: if $w \notin W$ then break
step 15: $A_m \leftarrow A_m \bigcup w$
step 16: $P_m \leftarrow P_m \cup w_{\text{out}}$
step 17: BP $\leftarrow$ BP$/(P_m \bigcap$BP)
step 18: BPM $\leftarrow$ BP
step 19: BP $\leftarrow$ BP$\bigcup(w_{\text{in}}/P_{m-1})$
step 20: $P_{m-1} \leftarrow P_{m-1} \cup w_{\text{in}}$
step 21: end while
step 22: BPQ $\leftarrow$ BPM$\bigcup$BPQ
step 23: If BPQ $\neq \emptyset$ then
step 24: 删除A_m层受BPQ影响的服务
step 25: end if
step 26: BP $\leftarrow$ BP/BPQ
step 27: BPH = BP
step 28: end for
step 29: while BP $\neq \emptyset$ do
step 30: 在规划图前面插入一个新的空P_{new}和A_{new}层
step 31: $P_{\text{new}} = P_0$/BP
step 32: while BP $\neq \emptyset$ do
step 33: 根据式(9)在W中选择w
step 34: if $w \notin W$ then break
step 35: $A_{\text{new}} \leftarrow A_{\text{new}} \bigcup w$
step 36: $P_0 \leftarrow P_0 \bigcup w_{\text{out}}$
step 37: BP $\leftarrow$ BP$/(P_0 \bigcap$BP)
step 38: BP $\leftarrow BP \bigcup(w_{\text{in}}/P_{\text{new}})$
step 39: end while
step 40: $A_0 \leftarrow A_{\text{new}}$
step 41: $P_0 \leftarrow P_{\text{new}}$
step 42: if BP $\neq \emptyset$ then break
step 43: BP$\bigcap$BPH $\neq \emptyset$ then break else 添加BP到BPH
step 44: end while
step 45: if BP $\neq \emptyset$ then按照规划图的反向搜索方法返回组合结果或者失败.

4 实验结果及其分析

为了验证本文提出的修复方法的性能,进行如下实验来模拟云边环境下规划图的修复和重新规划问题. 仿真在以下配置的计算机上进行:

1) CPU: Intel Core i5-10400F at 2.90 GHz;
2) 内存: 16 GB DDR4 2666 MHZ;
3) 硬盘: KINGSTON SNVS 500 GB;
4) 操作系统: Windows 10 Professional 64位.

实验中采用两个数据集[18]进行实验. 这些数据集的格式是基于本体、服务和服务编排的WSDL和OWL表示. 这些数据采用了语义Web服务的概念,其中使用接口和服务质量特征来描述Web服务. 数据集包含了大量的服务和输入(w_{in})/输出(w_{out})参数. 其中第1个数据集有572个服务和1 578个输入/输出参数; 第2个数据集有4 129个服务,超过12 388个输入/输出参数. 输入/输出参数对应于OWL中定义的类.

在模拟的云边环境中,设定5个云服务器和10个边缘服务器,之后将数据集中的服务按照一定顺序布置在这些服务器中. 此外设置3个评估标准进行比较: 用户从上传到得到结果的时间,解决方案中的服务数量和重新规划与修复方法对比原始计划的差

距(即与原始方案中不同服务的个数). 用户从上传数据到获得结果所需的时间包括了重新获得结果的时间开销(其中包括修复和重新规划过程的开销).

实验过程按照一定比例从服务集合中移除候选服务,在同一个服务损坏的环境中比较修复方法和重新规划方法得到结果所需的时间. 由图5可以发现,重新规划通常需要更多的时间来获得结果,而本文的修复方法(RDPGSC)以及原始的修复算法可以更快地得到服务组合结果. 并且RDPGSC只需要更少的时间,这是由于本文方法在修复过程中会优先选择响应时间更短的服务. 然而,随着服务被破坏比例的增加,两种方法所需的时间也会增加. 在较高的比例情况下,修复方法可能会失效,无法返回一个组合方案. 例如在数据集2中,如果删除超过12%的服务后的10次运行结果,则只有4次修复成功. 但通常情况下,修复方法可以更快地得到结果. 在数据集2中可以观察到当删除更多服务时,重新规划的时间略有减少,主要是因为随着可用候选服务的较少,问题规模也随之减小.

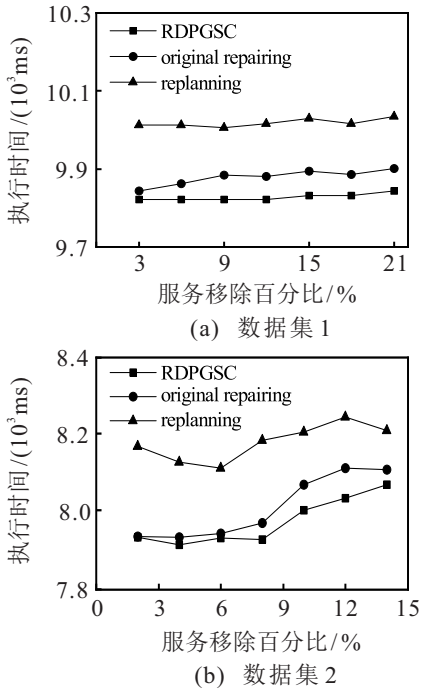

图5　获取解决方案的时间对比

图6显示了在修复规划图和重新生成规划图后解决方案中的服务数量. 可以看出,修复后产生的解决方案需要更少的服务,并且本文的方法与原始方法相比需要的服务也较少. 这里的解释是在修复过程中,本文在原始规划图的基础上进一步完善了规划图,因此修复的解决方案中的服务数量更接近原始方案. 这一点可以从图7中观察到,两种修复方法产生的解决方案比重新规划更接近原始规划图. 当然,随着移除服务比例的增加,无论是修复还是重新规划,与原始规划图都存在差距,这是可以理解的. 但是,修复也不会受限于原始规划图方案,并且所需的执行时间也会更少.

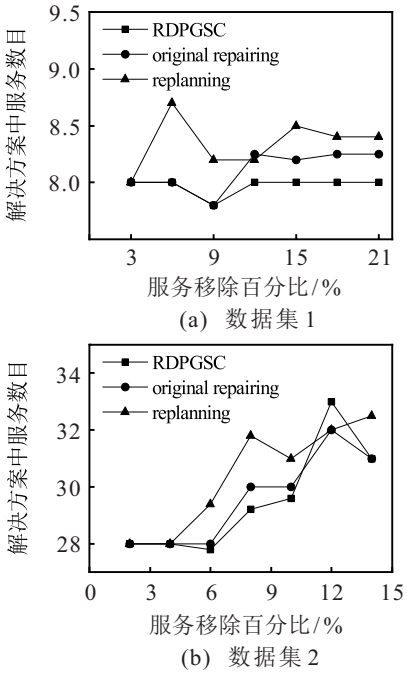

图6　获取解决方案中的任务数量的对比

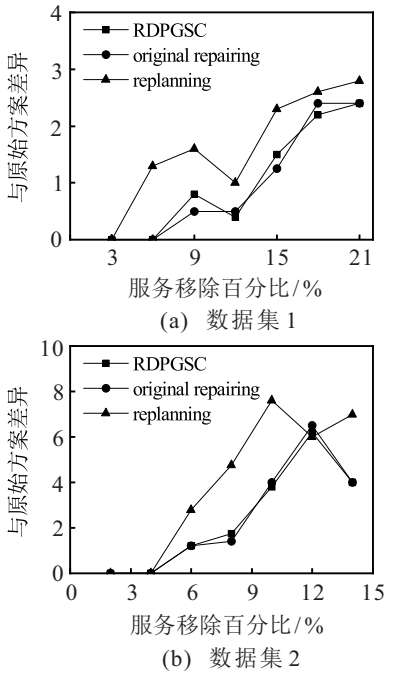

图7　获取解决方案与原始方案的差异

5　结　论

本文提出了一种基于规划图的云边服务组合修复方法,以应对在云边环境中面临的服务组合和规划挑战. 通过使用规划图来建模服务间的依赖关系和约束条件来满足用户需求. 修复算法通过搜索启发式函数快速修复部分规划图,直到满足所有输出条件

为止. 相比重新规划,修复方法能更快地找到服务组合解决方案,并最大限度地重用部分规划图内容.

实验证实了修复方法的优势,因此该方法能够满足用户需求并应对云边服务环境中的动态变化和不确定性. 未来研究可以进一步改进修复算法,提高修复效率和性能,并将其应用于更广泛的实际场景.

参考文献(References)

[1] Zhu M, Yu F L, Yan X K, et al. Scaling up mobile service selection in edge computing environment with cuckoo optimization algorithm[C]. 2021 IEEE International Conference on Services Computing. Chicago, 2021: 394-400.

[2] 李燕君, 蒋华同, 高美惠. 基于强化学习的边缘计算网络资源在线分配方法[J]. 控制与决策, 2022, 37(11): 2880-2886.
(Li Y J, Jiang H T, Gao M H. Reinforcement learning-based online resource allocation for edge computing network[J]. Control and Decision, 2022, 37(11): 2880-2886.)

[3] 吕灵灵, 杨志鹏, 张磊. 基于合约设计的移动边缘计算任务卸载策略研究[J]. 控制与决策, 2019, 34(11): 2366-2374.
(Lyu L L, Yang Z P, Zhang L. Contract theory based task offloading strategy of mobile edge computing[J]. Control and Decision, 2019, 34(11): 2366-2374.)

[4] Guo S T, Xiao B, Yang Y Y, et al. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing[C]. IEEE INFOCOM 2016 — The 35th Annual IEEE International Conference on Computer Communications. San Francisco, 2016: 1-9.

[5] 韩敏, 张丽君. 基于多样性检测的双子群多目标粒子群算法[J]. 控制与决策, 2017, 32(12): 2268-2272.
(Han M, Zhang L J. Bi-group multi-objective particle swarm optimization algorithm based on diversity metric[J]. Control and Decision, 2017, 32(12): 2268-2272.)

[6] Li J, Yan Y H, Lemire D. Scaling up web service composition with the skyline operator[C]. 2016 IEEE International Conference on Web Services. San Francisco, 2016: 147-154.

[7] Boussalia S R, Chaoui A. Optimizing QoS-based web services composition by using quantum inspired cuckoo search algorithm[C]. International Conference on Mobile Web and Information Systems. Cham: Springer, 2014: 41-55.

[8] 韩红桂, 徐子昂, 王晶晶. 基于Q学习的多任务多目标粒子群优化算法[J]. 控制与决策, 2023, 38(11): 3039-3047.
(Han H G, Xu Z A, Wang J J. A Q-learning-based multi-task multi-objective particle swarm optimization algorithm[J]. Control and Decision, 2023, 38(11): 3039-3047.)

[9] Li J, Yan Y H, Lemire D. Full solution indexing for top-K web service composition[J]. IEEE Transactions on Services Computing, 2018, 11(3): 521-533.

[10] Zhu M, Fan G D, Li J, et al. A service composition approach based on overall QoS and modified graphplan[J]. International Journal of Web and Grid Services, 2019, 15(4): 319-339.

[11] 韩敏, 段彦忠. 融合可信性评价的Web服务组合QoS优化[J]. 控制与决策, 2020, 35(8): 1859-1865.
(Han M, Duan Y Z. QoS optimization of Web services composition incorporating with credibility evaluation[J]. Control and Decision, 2020, 35(8): 1859-1865.)

[12] Friedrich G, Fugini M G, Mussi E, et al. Exception handling for repair in service-based processes[J]. IEEE Transactions on Software Engineering, 2010, 36(2): 198-215.

[13] Church J, Motro A. Recommending service repairs[C]. Proceedings of the 2014 IEEE 8th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems. New York, 2014: 11-18.

[14] Eder J, Mangler J, Mussi E, et al. Using stateful activities to facilitate monitoring and repair in workflow choreographies[C]. Proceedings of the 2009 Congress on Services — I. New York, 2009: 219-226.

[15] Yan Y H, Poizat P, Zhao L D. Repairing service compositions in a changing world[C]. Software Engineering Research, Management and Applications 2010. Berlin, 2010: 17-36.

[16] Yan Y H, Chen M. Anytime QoS-aware service composition over the GraphPlan[J]. Service Oriented Computing and Applications, 2015, 9(1): 1-19.

[17] Bettstetter C, Hartenstein H, Pérez-Costa X. Stochastic properties of the random waypoint mobility model: Epoch length, direction distribution, and cell change rate[C]. Proceedings of the 5th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems. New York, 2002: 7-14.

[18] WS-Challenge. Testsetgenerator2009[DB/OL]. (2009-08-09)[2023-08-15]. https://code.google.com/p/wsc-pku-tcs/downloads/list.

作者简介

高志浩(1998−), 男, 硕士生, 从事服务计算、服务选择等研究, E-mail: 21505020607@stumail.sdut.edu.cn;

李静(1986−), 女, 副教授, 博士, 从事服务计算、计算机网络等研究, E-mail: li_jing@sdut.edu.cn;

祝铭(1983−), 男, 讲师, 博士, 从事服务计算、边缘计算等研究, E-mail: zhu_ming@sdut.edu.cn;

刘彤阳(1999−), 男, 硕士生, 从事服务计算的研究, E-mail: liutongyang0716@163.com;

鲁瑞(1998−), 男, 硕士生, 从事计算机视觉的研究, E-mail: u7353434@anu.edu.au.