哈爾濱工業大學

本科毕业论文（ 设计 ） 中期报告

题　目：面向公平性的云边微服务系统部署方法

专　　业　软件工程

学　　生　付书煜

学　　号　2021111824

指导教师　贺祥

日　　期　2025 年 3 月 日

哈尔滨工业大学教务处制

目 录

1 论文（设计）工作是否按开题报告预定的内容及进度安排进行

我的研究工作基本按照开题报告中的规划进行。截至目前，已完成系统建模、算法实现，并完成了部分实验。具体工作进度安排如表 1-1 所示：

表 1-1 工作进度安排表

工作安排	周数	起止时间
系统建模与设计阶段	3	2024.11.20-2024.12.15
算法设计与实现	5	2024.12.16-2025.01.19
仿真实验与数据收集	4	2025.01.20-2025.02.16
敏感性分析与模型调优	3	2025.02.17-2025.03.09
成本效益分析与优化验证	4	2025.03.10-2025.04.06
结题，论文编写	2	2025.04.07-2025.04.20

2 已完成的研究工作及成果

在智能网络服务的实际应用中，我们的用户群体分布在不同的地理位置，既包括城市中心的用户，也包括偏远地区的用户，他们通过各种终端设备访问系统。为了保证高效的服务质量，我们采用云-边协同计算架构，在多个地理位置部署边缘服务器，并结合计算资源丰富的云服务器提供支持。然而，在这一架构下，如何确保不同用户在资源受限的环境下获得公平的服务体验成为了核心挑战。

首先，由于地理位置的不同，用户的网络条件存在较大差异。距离边缘服务器较近的用户可以享受较低的传输延迟，而远离边缘服务器的用户则可能面临较长的网络延迟，导致服务体验存在显著不均衡。其次，用户的优先级也是影响公平性的关键因素。高优先级用户（如 VIP 用户或付费用户）通常享有更多计算资源，从而获得更短的响应时间，而普通用户的请求在资源紧张时可能会被延迟甚至受限。这种优先级差异虽然可以提升高付费用户的体验，但如果资源分配不合理，可能会导致普通用户的服务质量下降过多，使整个系统的公平性变差。此外，资源分配问题进一步加剧了公平性挑战。边缘服务器的计算能力有限，若大量用户请求集中在某些边缘节点，可能会造成资源超载，而云服务器虽然资源丰富，但网络传输延迟较高，无法完全替代边缘计算。

我们的目标是在边缘服务器资源有限的前提下，通过优化资源分配和服务实例的部署，使得不同优先级的用户都能够在合理的响应时间内获得服务，尽可能提高系统的公平性。同时，我们还需要确保云边协同架构在部署实例时能够平衡成本和资源消耗，避免不必要的资源浪费或过度负载，从而实现系统的高效运行和经济可行性。

为了实现这一目标，我的研究工作从以下几个方面展开：云-边协同微服务系统的设计与建模、算法实现、实验相关平台开发及实验。接下来，我将详细介绍我已完成的研究工作及所取得的成果。

2.1 云-边协同微服务系统的设计与建模

2.1.1 场景描述

我们的云-边协同微服务系统由一组服务实例组成，这些服务实例部署在不同的边缘服务器以及云服务器上，为用户提供计算服务。用户分布在不同的地理位置，并根据付费水平等因素被赋予不同的优先级，从而享有相应的计算资源。每个用户的请求数据需在满足计算资源约束的前提下，由合适的服务器进行处理。系统通过优化连接关系，合理分配云-边资源，以确保不同用户的服务公平性，同时控制部署成本，防止边缘节点超载或云端延迟过高，从而实现高效的云-边协同计算。

2.1.2 变量定义

为了建立云-边协同计算模型，我们定义如下变量：

2.1.2.1 （定义一）用户

假设共有 n 个用户要接入到我们的系统获取服务，定义用户集为 $U = \{u_1, u_2, \cdots, u_n\}$，每个用户 u_i 被表示为一个多维变量：$u_i = < loc_i, Q_i, W_i, D_i >$。

其中，$loc_i = (x_i, y_i)$ 示用户所处的地理位置，其会影响用户连接到服务器的选择以及传输延迟计算；Q_i 是系统根据用户的重要程度（如付费用户、普通用户等）来为用户赋予的变量值，Q_i 值越高，代表该用户的重要性越大。此变量通常由业务策略或平台规则定义，而非用户自行决定；W_i 为用户权重，用于衡量用户响应时间对公平性指标（加权 Jain 指数）的影响。系统依据 Q_i 分配 W_i，通常 W_i 与 Q_i 成正比，以确保高优先级用户在公平性计算中占据更大的比重。D_i 表示用户 u_i 发往服务器（云或边缘）的数据量，该值取决于用户实际的计算需求，直接影响服务器计算资源分配、传输延迟及整体响应时间。

2.1.2.2 （定义二）服务器

系统中的服务器包括边缘服务器和云服务器，我们用集合 S 表示所有服务器：$S = S_{edge} \cup S_{cloud}$，其中 S_{edge} 代表所有的边缘服务器，S_{cloud} 代表所有的云服务器。每个服务器 s_j 由以下变量描述：$s_j = < loc_j, R_j^{server}, P_j, b_j, c_j, T >$。

其中，$loc_j = (x_j, y_j)$ 表示服务器的地理位置，其影响用户连接的选择及传输延迟计算；$R_j^{server} = \{r_j^{cpu\_total}, r_j^{mem\_total}, r_j^{b\_total}\}$ 表示服务器的 CPU、内存以及带宽总量；P_j 代表服务器的总计算能力，决定了其对用户请求的处理效率，通常情况下，云服务器的处理能力要强于边缘服务器；b_j 为服务器提供的可用带宽，影响数据传输时延；c_j 为该服务器的运行成本，包括计算资源消耗和网络流量费用；T 为延迟矩阵，其中记录了连接到服务器 s_j 的用户 u_i 获得服务响应的时延 t_{ij}，t_{ij} 由传输时延 $t_{trans_{ij}}$ 和处理时延 $t_{proc_{ij}}$ 两部分组成。

2.1.2.3 （定义三）连接关系与资源分配

用户与服务器之间的连接关系用二进制变量 x_{ij} 来表示。x_{ij} 为 1 表示用户 u_i 连接到服务器 s_j，为 0 则表示没有连接。每个服务器上可能部署了一个或多个服务实例；如果没有用户连接到某个服务器，该服务器可以不部署任何服务实例。用户 u_i 连接到服务器 s_j 时，s_j 需要分配给用户一定的资源，用集合 $R_i^{user} = \{r_i^{cpu}, r_i^{mem}, r_i^{b}\}$ 来表示。其中 $r_i^{cpu} = f_{cpu}(D_i) \cdot W_i$，$r_i^{mem} = f_{mem}(D_i) \cdot W_i$，$r_i^{b} = f_b(D_i) \cdot W_i$，分别表示用户所需的 CPU、内存以及带宽资源

量；$f_x(D_i)$ 为用户数据请求量 D_i 与 CPU、内存以及带宽等资源分配量的转换函数，其根据实际转换情况进行设置；W_i 为用户权重，使得高优先级用户能获取更多资源，体现了模型对不同优先级用户资源分配的差异化处理。

2.1.3 约束条件

（1）**平均响应时间约束** 确保每个优先级类别用户的平均响应时间不会超过设定的上限，如公式（2-1）所示：

$$\frac{1}{|U_{Q_i}|}\sum_{u_j \in U_{Q_i}} \sum_{s_k \in S} x_{jk} \cdot t_{jk} \le T_{Q_i}^{max}, \ \forall Q_i \tag{2-1}$$

其中，$T_{Q_i}^{max}$表示优先级为 Q_i 的用户的平均响应时间上限。

（2）**边缘节点资源限制** 每个边缘节点 s_j 的资源消耗不得超过其最大可用资源，约束公式如下：

$$\sum_{i=1}^{n} x_{ij} \cdot r_i^{cpu} \le R_j^{cpu\_total}, \forall s_j \in S_{edge} \tag{2-2}$$

$$\sum_{i=1}^{n} x_{ij} \cdot r_i^{mem} \le R_j^{mem\_total}, \forall s_j \in S_{edge} \tag{2-3}$$

$$\sum_{i=1}^{n} x_{ij} \cdot r_i^{b} \le R_j^{b\_totsl}, \forall s_j \in S_{edge} \tag{2-4}$$

（3）**部署成本限制** 边缘节点与云节点的部署成本总和不得超过整体预算。

$$C_{total} = \sum_{s_j \in S_{edge}} c_j^e + \sum_{s_j \in S_{cloud}} c_j^c \le C_{max} \tag{2-5}$$

其中，C_{max} 为设定的系统的部署成本上限。

（4）**用户与服务器连接限制** 每个用户 u_i 必须连接到唯一一个服务器。

$$\sum_{j=1}^{m} x_{ij} = 1, \forall u_i \in U \tag{2-6}$$

2.1.4 优化目标

（1）**公平性目标函数** 用 f 表示，即 $f = \max(F_{Jain})$，其中 F_{Jain} 为加权 Jain 公平性指数，其定义为公式（2-7）：

$$F_{Jain} = \frac{\left(\sum_{i=1}^{n} t_{ij}^{weight}\right)^2}{n \cdot \sum_{i=1}^{n} \left(t_{ij}^{weight}\right)^2} \tag{2-7}$$

公式中，分子为所有用户加权响应时间总和的平方，反映了加权时间的集中程度，若各用户的加权响应时间较为相近，总和会较大，平方后的值也更大，这表明用户间响应时间差异较小；分母是所有用户加权响应时间的平方和，体现了系统中所有用户响应时间的散布程度，若各用户响应时间相差较大，这个和将会较大，意味着公平性较差。加权响应时间能起到调节公平性的作用，若高权重用户的响应时间更短，其加权响应时间会更接近低权重用户的加权响应时间，使分子总和更集中，差异减小；同时会减少分母中高权重用户平方项的贡献，让整体分母变小。

整个目标函数通过最大化 F_{Jain} 来优化公平性，当 趋近于 1 时，代表所有用户的加权响应时间几乎完全相同，系统达到公平状态；当 F_{Jain} 趋近于 0 时，说明加权响应时间差异非常大，系统不公平。该目标函数不仅可用于公平性评估，还能通过引入加权响应时间，根据用户优先级调节不同用户之间的响应时间差异，保证不同优先级用户的响应时间公平。

2.2 算法实现

2.2.1 算法实现思路

在本研究中，我们主要采用了基于贪心[1]的优化方法来解决云-边协同架构中的公平性服务部署问题。该算法基于用户的优先级，逐一为每个用户分配服务器上的资源，以最大化系统的公平性。核心思路是每次选择一个局部最优解（即为当前用户选择能使 Jain 公平性指数最大的服务器），以此逐步优化系统的整体性能。该方法通过优化加权 Jain 公平性指数来实现公平性目标，同时在每一步中确保资源分配满足服务器资源的约束。

2.2.2 算法具体流程

（1）**初始化**　初始化用户到服务器的分配矩阵为零矩阵，并设置每个服务器的资源使用情况。

（2）**用户优先级排序**　根据用户的优先级（例如，高付费用户优先）将所有用户进行排序。优先级较高的用户将在分配资源时得到优先考虑。

（3）**分配服务器资源**　根据每个用户的需求，从可用的服务器中选择一个最适合的服务器进行资源分配。该选择过程旨在最大化加权 Jain 公平性指数，同时满足服务器的资源限制。

（4）**更新资源使用情况**　每分配一次资源，更新相应服务器的资源使用情况，确保不会超过服务器的最大资源限制。

（5）**检查约束条件**　在每一步分配之后，检查当前资源分配是否满足所有约束条件，包括服务器负载、部署成本以及响应时间上限等。如果所有条件满足，则继续分配下一用户的资源。

（6）**退出条件**　如果成功为所有用户分配服务器并满足所有约束，返回最终的分配矩阵；若在多次尝试后仍未找到满足约束条件的结果，则退出并给出警告。

2.2.3 算法伪代码

基于贪心的优化方法的伪代码如下所示：

Greedy User-Server Allocation

Input: U ← user set, S ← server set, resourceLimits, maxAttempts
Output: X ← connection matrix

```
X ← initializeZeroMatrix(U, S)
valid ← false
attempt ← 0
while not valid and attempt < maxAttempts do
    X ← resetMatrixAndResources(U, S)
    for each user u in U do
        bestServer ← -1
        for each server s in S do
            tempAssign(u, s)
```

```
            if checkResourceConstraints(s, u, resourceLimits) then
                jainIndex ← calculateJainIndex(X, S)
                if jainIndex > bestJain then bestJain ← jainIndex, bestServer ← s
            end if
            undoTempAssign(u, s)
        end for
        if bestServer ≠ -1 then assignUserToServer(X, u, bestServer)
    end for
    if checkAllConstraints(X, S, resourceLimits) then valid ← true
    attempt ← attempt + 1
end while
if attempt ≥ maxAttempts then Warning
return X
```

2.2.4 对比算法

为了验证基于贪心的优化方法的有效性，本研究将其与几种其他优化算法进行了对比：

（1）**优化求解器（Gurobi）** Gurobi [2] 使用数学优化工具对资源分配进行全局优化，目标是最大化系统的公平性并满足约束条件。优化求解器能够找到理论上的最优解，但计算开销较大，尤其在大规模问题中。

（2）**遗传算法（GA）** 遗传算法[3,4]是一种基于自然选择和遗传学原理的启发式优化算法。通过模拟自然选择过程，GA 能够在大规模问题中找到近似最优解，但相比于贪心算法，计算时间较长。

（3）**无公平性算法** 此算法以最小化平均响应时间为优化目标，不考虑用户之间的公平性。它可以在某些情况下降低用户响应时间，但由于忽视了优先级和资源分配的差异，将无法确保系统公平性，在本实验中主要是用来对照。

2.3 实验平台开发

为了支持云-边协同计算系统的研究，特别是在优化公平性和资源分配方面，我参与了 MicroForge 平台前端部分的开发，主要负责数据源管理模块的实现。以下是这该模块的设计与实现细节。

2.3.1 功能描述

数据源管理模块提供了一个界面，用于展示和管理系统中的所有数据源。每个数据源都包括其基本信息、数据类型、资源使用情况（如 CPU、内存、带宽），以及实时状态监控功能。用户可以通过此模块：查看每个数据源的当前资源使用状态、实时监控各数据源的负载和资源消耗情况、查询并管理数据源，确保系统能够根据实际负载动态调整资源分配。

2.3.2 需求分析

（1）**数据源展示** 需要提供每个数据源的基本信息（如名称、类型、描述等）。

（2）**资源使用情况展示**　包括每个数据源的 CPU、内存、带宽等资源的实时使用情况。

（3）**实时更新**　支持系统实时监控和更新数据源的资源使用情况，确保能够动态响应。

（4）**据源查询与管理**　允许用户对数据源进行增删改查操作。

2.3.3 设计与实现

（1）**界面设计与交互**

a. 采用表格展示每个数据源的基本信息，包括数据名称、类型等情况。

b. 用户可以通过下拉菜单、输入框等方式进行数据源的筛选、搜索与管理。

c. 提供详细信息查看按钮，支持展开数据源中各类资源的更多属性信息。

（2）**数据更新与交互**

a. 后端数据通过 RESTful API 与前端交互，实现数据的增删改查。

b. 结合 WebSocket 技术，前端可以实时接收资源使用情况的变化，确保数据动态更新。

c. 使用 Redux 进行状态管理，优化数据存储和共享，提高响应效率。

2.3.4 模块功能流程

图 2-1 展示了数据源管理模块的运行流程。

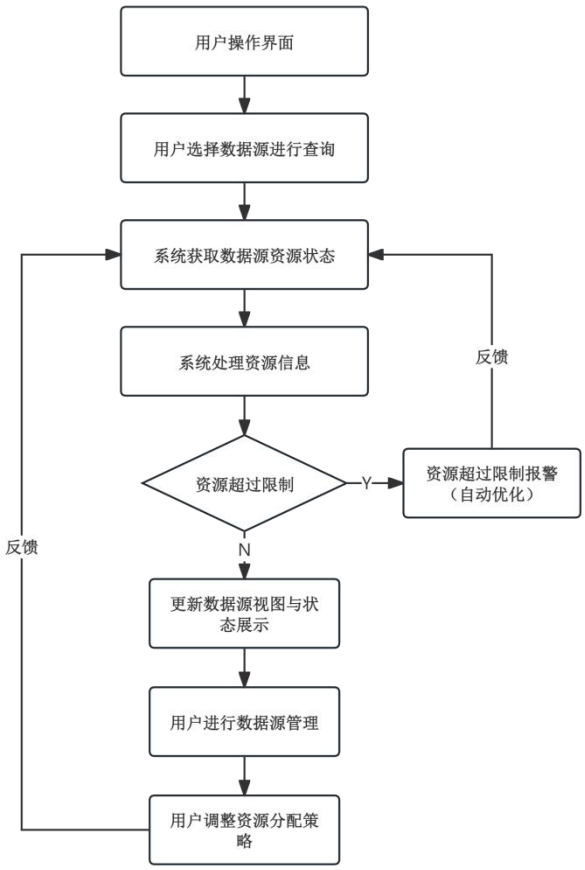


图 2-1 数据源管理模块功能流程

2.3.5 模块展示

图 2-2 (a)、2-2 (b)、2-2 (c) 分别展示了平台上数据源管理模块的不同功能页面。

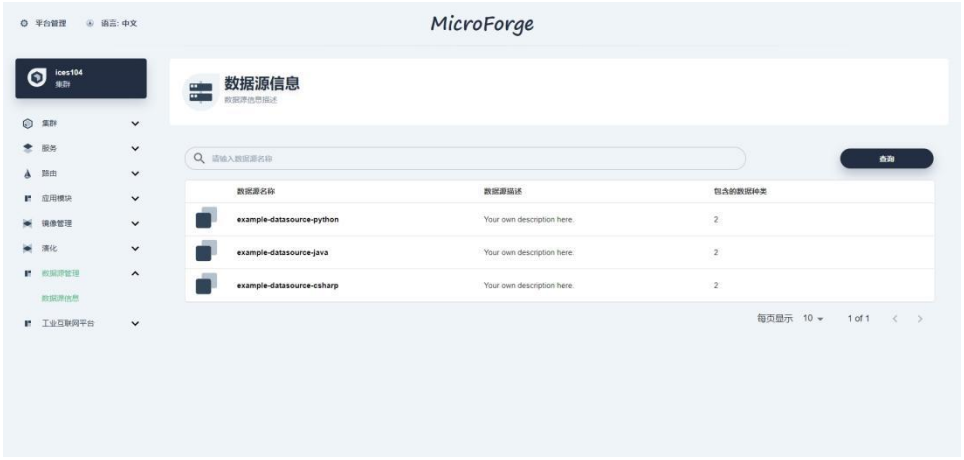


图 2-2 (a) 数据源信息展示页面

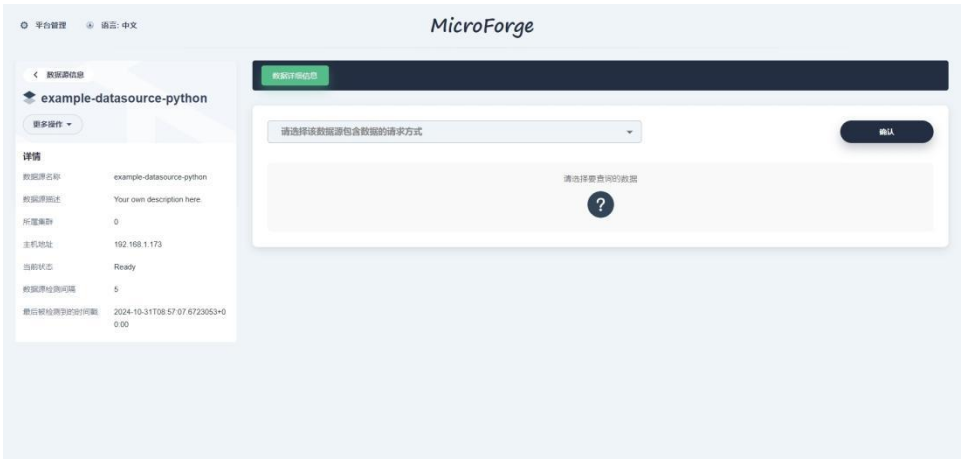


图 2-2 (b) 资源信息获取页面

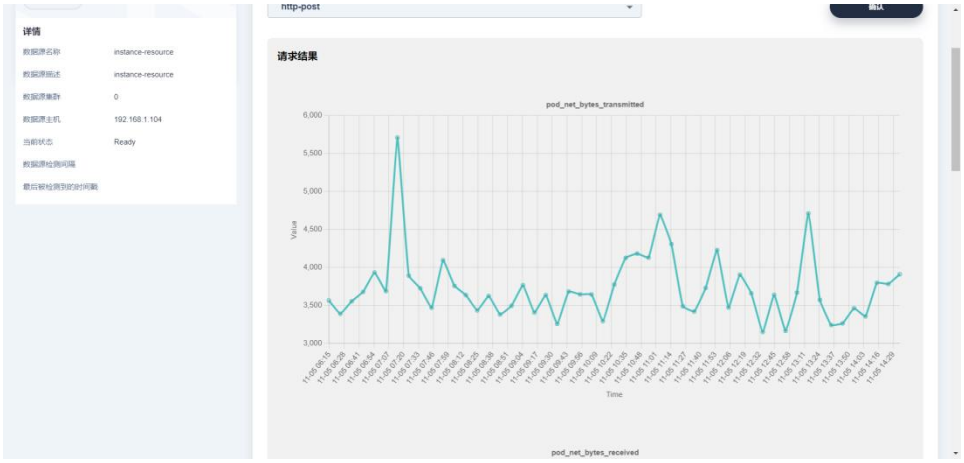


图 2-2 (c) 资源信息展示页面

2.3.6 平台功能模块与本研究的结合

数据源管理模块帮助监控云-边协同系统中的服务器各类资源使用情况，为模型的资源分配和性能评估提供关键的数据支持。研究人员可以实时获取各服务器的资源状态，优化

资源分配策略，并对高优先级用户进行更精确的资源分配。

通过该模块，MicroForge 平台为本研究提供了一个灵活且强大的实验环境，使得不同网络条件和资源使用情况下的算法优化与测试变得更加高效和直观。

2.4 实验

2.4.1 实验设计

为了验证所提出的云-边协同系统在不同网络环境下的性能表现以及资源分配算法的有效性，本研究设计了一系列实验。实验的主要目的是评估不同算法在处理不同用户数量和优先级时的平均响应时间和公平性表现。

实验设置如下：

（1）**用户数量** 选择 100、150 和 200 个用户，来模拟不同负载下的用户场景。

（2）**用户优先级设置** 为每个用户分配一个优先级，本实验中包括优先级 1、2、3 三个类别，以测试在优先级差异下的公平性。

（3）**算法选择** 实验中使用四种算法进行对比，包括基于贪心的优化方法、Gurobi 优化求解器、遗传算法（GA）以及无公平性的算法。

（4）**实验平台** 本实验应用到了 MicroForge 平台，平台的集群网络、数据源管理模块为实验提供了必要的实验环境。

2.4.2 实验过程

（1）**初始化阶段** 为每个实验配置用户数量、优先级以及相应的服务请求。并初始化每个算法的输入参数。

（2）**算法执行** 根据所选算法，执行资源调度与服务实例部署过程。每个算法根据用户的优先级、请求大小和网络延迟等条件进行资源分配。

（3）**数据采集与分析** 收集每个用户的响应时间、系统资源利用率等数据，并进行统计分析。重点关注不同优先级用户的平均响应时间差异，以及不同算法在公平性优化方面的表现。

（4）**实验验证** 通过实验验证模型和算法的正确性，并对不同算法的性能进行对比分析，特别是在平均响应时间和公平性方面的差异。

2.4.3 实验结果与分析

2.4.3.1 实验结果展示

在本部分中，我将以 200 个用户为例，展示各算法的运行结果。

（1）**用户与服务器初始分布** 图 2-3 展示了用户与边缘服务器和云服务器的初始分布情况。

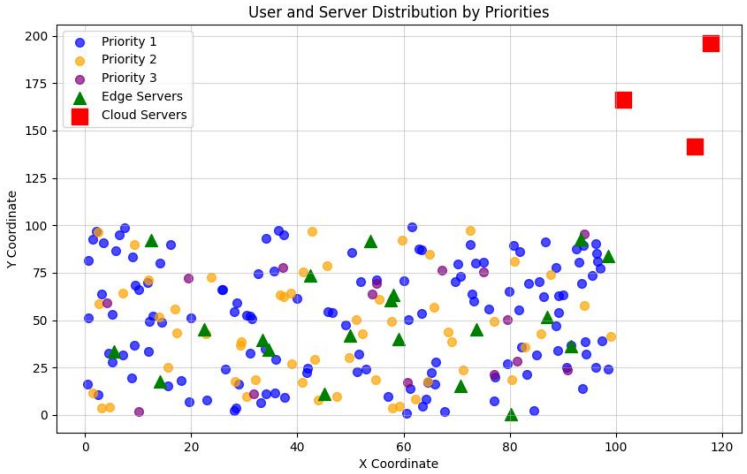
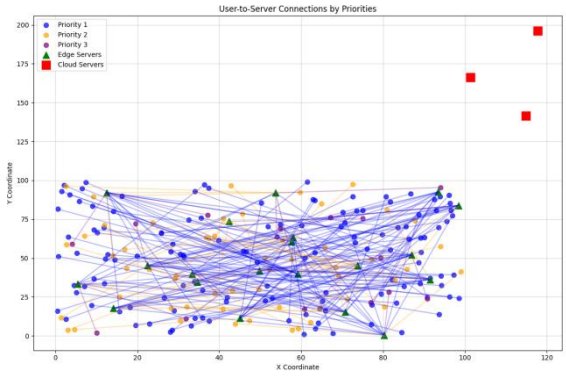
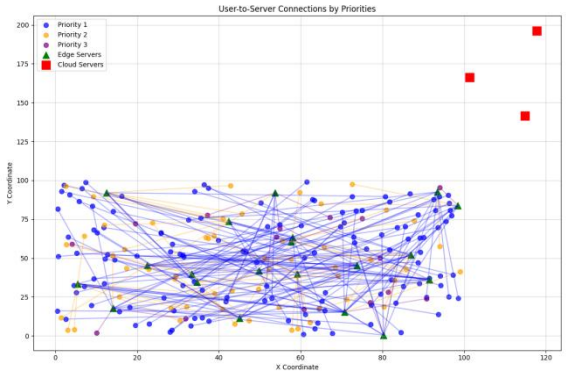


图 2-3 用户与服务器的初始分布

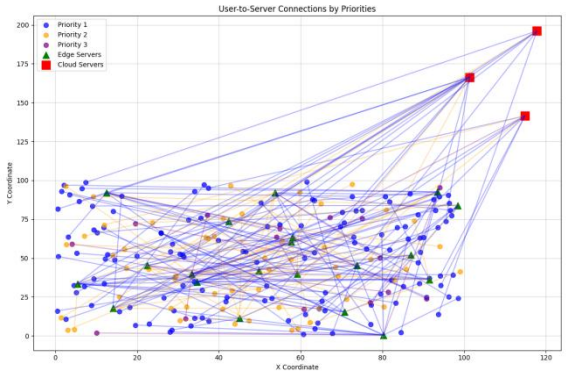
（2）**用户与服务器连接情况** 图 2-4 (a)、2-4 (b)、2-4 (c) 、2-4 (d) 分别展示了基于贪心的优化方法、Gurobi 优化求解器、遗传算法（GA）以及无公平性的算法运行后的用户与服务器连接情况。



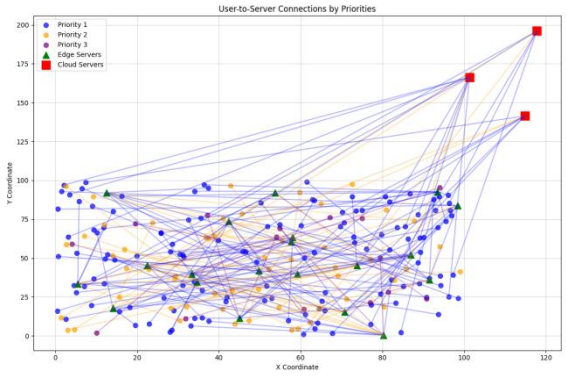
(a) 基于贪心的优化方法



(b) Gurobi 优化求解器



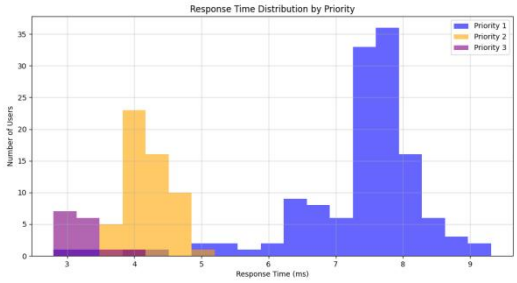
(c) GA 算法



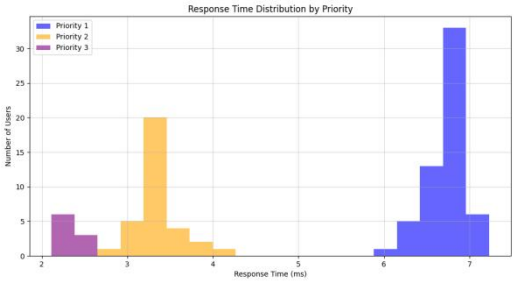
(d) 无公平性算法

图 2-4 用户与服务器连接情况

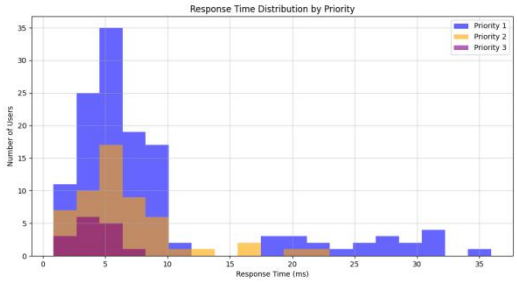
（3）**各优先级用户的响应时间分布** 图 2-5 (a)、2-5 (b)、2-5 (c) 、2-5 (d) 分别展示了基于贪心的优化方法、Gurobi 优化求解器、遗传算法（GA）以及无公平性的算法运行后的各个优先级用户的响应时间分布情况。



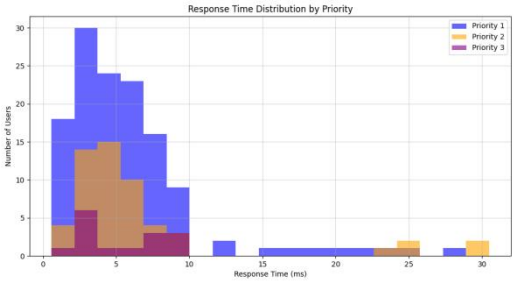
(a) 基于贪心的优化方法



(b) Gurobi 优化求解器



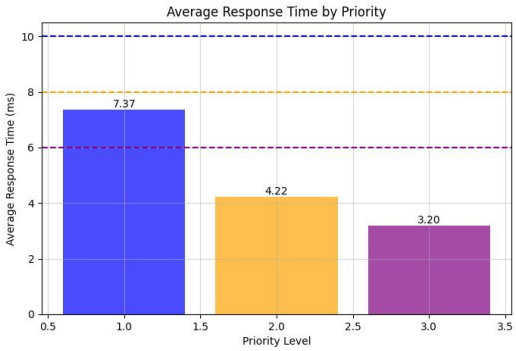
(c) GA 算法



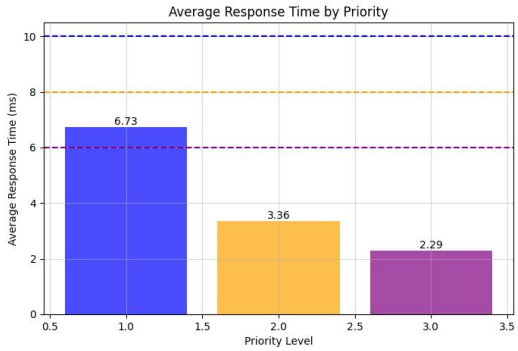
(d) 无公平性算法

图 2-5 各优先级用户的响应时间分布情况

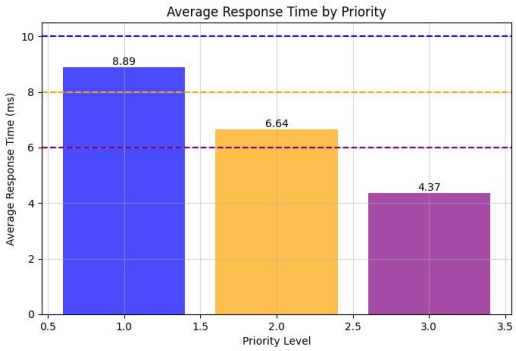
（4）**各个优先级用户的平均响应时间** 图 2-6 (a)、2-6 (b)、2-6 (c) 、2-6(d) 分别展示了基于贪心的优化方法、Gurobi 优化求解器、遗传算法（GA）以及无公平性的算法运行后的各个优先级用户的平均响应时间。



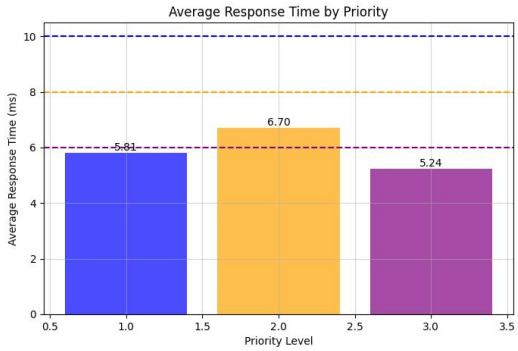
(a) 基于贪心的优化方法



(b) Gurobi 优化求解器



(c) GA 算法



(d) 无公平性算法

图 2-6 各优先级用户的平均响应时间

2.4.3.2 实验结果分析

在本部分中，我将展示不同算法在不同用户数量和优先级设置下的性能表现，具体从不同优先级用户平均响应时间和系统公平性表现等方面进行分析。

（1）**不同用户数量下各算法的平均响应时间** 图 2-7 (a)、2-7 (b)、2-7 (c) 分别展示了用户数量为 100、150 和 200 时，各算法不同优先级用户的平均响应时间。

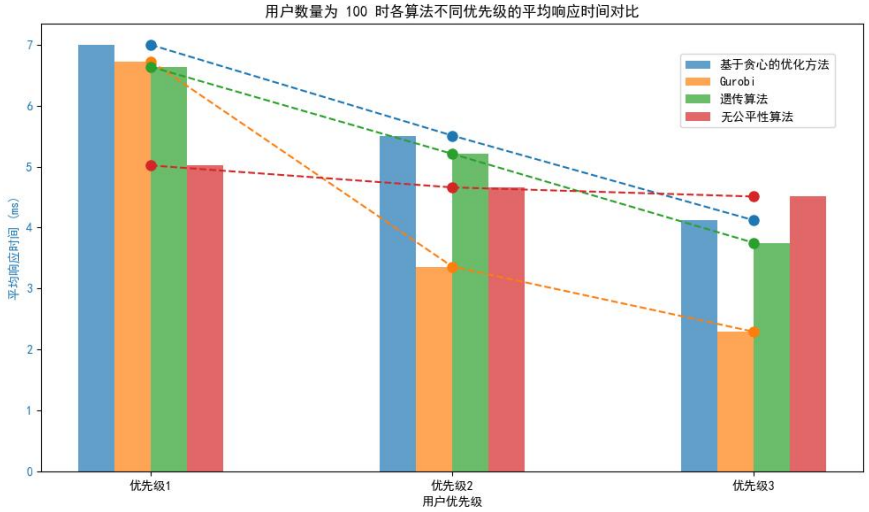


图 2-7 (a) 用户数量为 100 时各算法平均响应时间对比

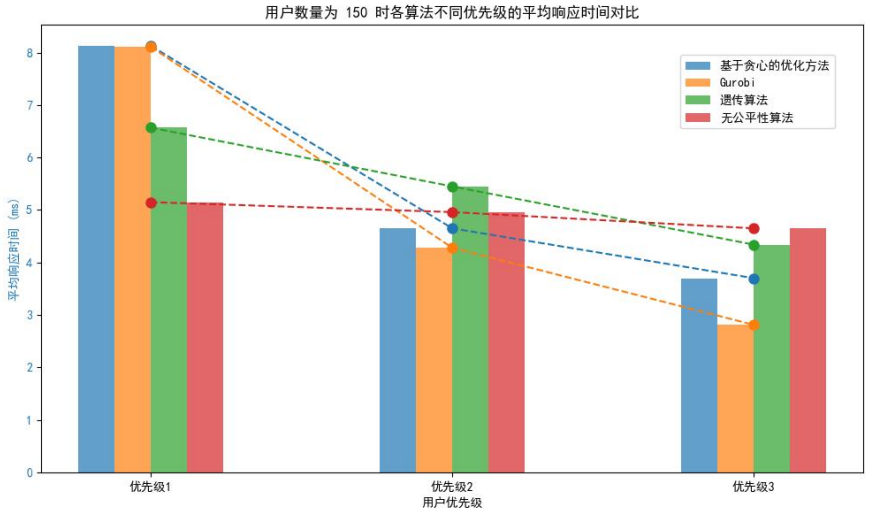


图 2-7 (b) 用户数量为 150 时各算法平均响应时间对比

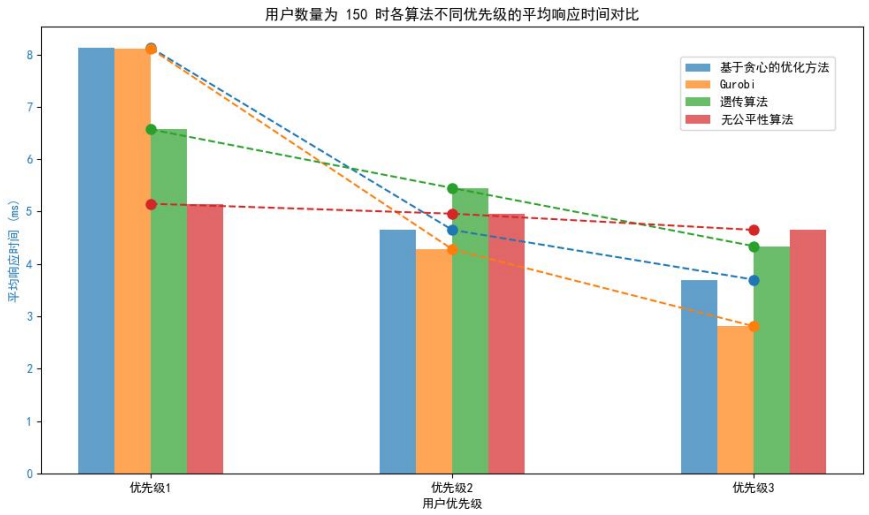


图 2-7 (c) 用户数量为 200 时各算法平均响应时间对比

我从以下两个方面对仿真结果进行分析：

首先，是在同一算法下不同优先级用户的平均响应时间情况。理论上，在资源有限的情况下优先级越高的用户应优先获得资源，因此他们的响应时间应该更短。实验结果表明，基于贪心的优化方法和 Gurobi 优化算法能够有效根据优先级分配资源，确保高优先级用户的响应时间明显较短，而低优先级用户的响应时间则较长。相比之下，无公平性算法没有考虑优先级差异，导致所有用户的响应时间趋于一致，无法有效区分高低优先级用户，影响高优先级用户的体验。

其次，四种算法的对比结果显示，基于贪心的优化方法在资源分配方面表现较为均衡，尤其在大规模用户场景下，能够稳定保持低响应时间。Gurobi 优化求解器虽然提供最优解，但计算复杂度较高。遗传算法在小规模场景表现良好，但在用户数量增多时响应时间波动较大，且计算时间较长。无公平性算法忽视了优先级差异，尽管减少了平均响应时间，但无法保证公平性。总的来说，基于贪心的优化方法在平衡公平性和响应时间方面表现最佳，尤其适合大规模用户场景。

（2）**不同用户数量下各算法的加权 Jain 指数** 图 2-8 展示了不同用户数量下，各算法的加权 Jain 指数，该指数用于衡量系统的公平性，值越大表示系统的公平性越好。

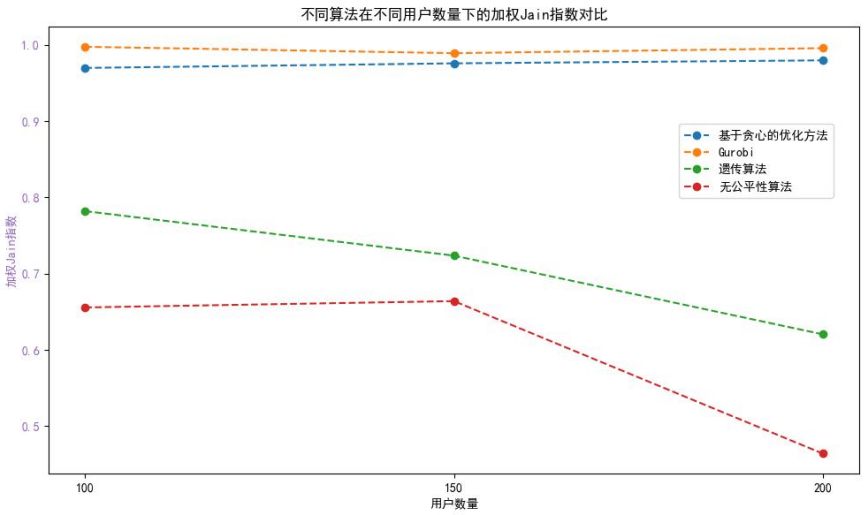


图 2-8　不同用户数量下各算法的加权 Jain 指数

从实验结果可以看出，随着用户数量的增加，所有算法的加权 Jain 指数均有所下降，表明系统的公平性在用户数量增多时有所降低。然而，基于贪心的优化方法和 Gurobi 优化求解器在所有用户数量下的加权 Jain 指数始终较高，显示出较好的公平性，尤其在大规模用户场景中，二者的公平性明显优于其他算法，说明它们能提供更优化的资源分配方案。尽管如此，由于基于贪心的优化方法的实现简单且计算开销较低，它在实际应用中表现更为优越，特别是在资源有限的情况下。相比之下，无公平性算法的加权 Jain 指数最低，表明该算法未能考虑用户优先级和公平性，导致不同用户之间的响应时间差异较大。总体来看，基于贪心的优化方法在平衡实现简单性和系统公平性方面表现突出，特别适合应用于大规模用户场景。

3 后期拟完成的研究工作及进度安排

后期工作将按以下进度安排完成：

（1）**模型调优与验证**（2025.03.10-2025.03.31）　进一步对现有模型进行验证，评估模型在不同环境下的表现，确保其具备实际应用的可行性。

（2）**算法优化**（2025.04.01-2025.04.30）　优化算法性能，提升算法的计算效率和稳定性，确保大规模问题处理的可行性。

（3）**论文撰写与总结**（2025.04.01-2025.04.30）　总结研究成果，并完成论文的编写与完善。

4 存在的问题与困难

（1）**实验数据的收集与分析困难**　由于实验需要在实际的云边协同环境中运行，数据收集面临一定的困难。尤其是在多种网络环境和设备条件下进行实验时，数据的准确性和代表性可能会受到不同网络延迟、带宽波动以及设备性能差异的影响。尽管通过仿真实验模拟了部分场景，但实际环境中的动态变化仍然是一个需要克服的问题。此外，由于平台的复杂性和多样性，实验数据的整合和分析也存在一定的难度。

（2）**公平性评估的主观性**　在公平性优化过程中，不同用户的需求差异较大，如何平衡这些差异并避免过于主观的判断，仍是一个挑战。

5 论文按时完成的可能性

根据目前的进度安排，研究工作和论文编写的各个环节都在按计划推进。系统建模、算法设计以及部分仿真实验已经完成，且模型调优和成本效益分析也已规划到位。在接下来的两个月内，重点将放在完成算法优化和模型验证上，以确保系统的实际应用可行性和稳定性。所有的实验和算法优化工作预计将在 3 月底完成，论文撰写阶段将从 4 月开始并顺利进入总结和完善阶段。

尽管存在一些实验数据收集和公平性评估的挑战，但通过已有的研究成果和数据支持，项目按时完成的可能性较高。在接下来的时间内，确保顺利完成剩余的工作，应该能够按计划完成毕业论文的撰写和提交。

6 主要参考文献

[1] Jungnickel D, Jungnickel D. The greedy algorithm[J]. Graphs, networks and algorithms,1999: 129-153.

[2] Pedroso J P. Optimization with gurobi and python[J]. INESC Porto and Universidade do Porto, Portugal, 2011, 1.

[3] Reeves C, Rowe J E. Genetic algorithms: principles and perspectives: a guide to GA theory[M]. Springer Science & Business Media, 2002.

[4] 葛继科，邱玉辉，吴春明，等. 遗传算法研究综述[J]. 计算机应用研究, 2008, 25(010): 2911-2916.