

本科生毕业设计（学术论文）



题 目 **基于区块链的分布式算力汇聚方法**

学 院  **软件学院**

专 业  **软件工程**

学生姓名  **周林**

学 号  **2015141463261** 年级 **2015**

指导教师  **郭兵**

教务处制表

二Ο一九年五月三日

**基于区块链的分布式算力汇聚方法**

软件工程

学生 周林 指导老师 郭兵

**[摘要]** 摘要应以概括地、不加注释地摘录本论文的研究目的、方法、结果和结论；或简洁的介绍本论文阐述的主要内容及取得的进展。客观反映原文内容，不得简单地重复标题名中已有的信息，要着重反映论文的新内容和特别强调的观点，以便读者能够很容易地知道文章的基本内容。摘要应具有独立性和自含性，不应出现图表、冗长的数学公式和非公知公用的符号、缩略语。不必要的词语，诸如副词和形容词、In this paper等，尽可能省去；背景信息、知识介绍、中间实验数据可以不要（结论性的实验数据当然还是要的）；不要说别人的缺点；过去研究的细节不要过长，可一笔带过；今后的计划不要写进去。摘要宜采用第三人称过去式的写法尽量用纯文字叙述。（如“对……进行了研究”，“综述了……”等；不应写成“本文”，“我校……”等）。摘要是提供给文献检索数据库做重复传播的情报资料，应该是一篇简明扼要的短文，一般在500～600字左右，概括介绍（1）主要研究背景，即从事这项工作的目的和重要性，（2）主要的研究内容，（3）采用的主要方法，（4）获得的主要结论，给读者一个对本论文的总体、初步的了解和认识。

**[主题词]** 区块链；合同网算法；机器学习；分布式；算力汇聚

**Distributed Power Convergence Method Based on Block Chain**

Software Engineering

Student： ZHOU Lin Adviser: GUO Bing

**[Abstract]** □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.

**[Key Words]** □□□□□；□□□□□；□□□□□；□□□□；□□□□□□；□□□□□□；□□□□□；□□□□□□。

**目 录**

1 绪论 5

1.1 研究背景 5

1.2 国内外研究现状 5

1.3 论文主要工作 7

1.4 论文组织与结构 7

2 算法基础简介 8

2.1 Agent技术 8

2.2 分布式机器学习技术 8

2.2.1 机器学习关键技术 8

2.2.2 logistics回归 9

2.2.3 数据并行和模型并行 9

2.3 本章小结 10

3 算法解决方案 10

3.1 相关技术 10

3.1.1 合同网模型 10

3.1.2 黑板模型 11

3.1.3 Paxos算法 12

3.1.4 Raft算法 13

3.1.5 总结 13

3.2 改进合同网模型 13

3.2.1 改进思想 13

3.2.2 数学建模 15

3.2.3 详细设计 16

3.3 本章小结 19

4 改进合同网模型实验验证 20

4.1 实验环境 20

4.2 实验设计 20

4.3 实验结果与分析 22

4.3.1 实验过程数据 22

4.3.2 实验结果分析 24

4.4 本章小结 26

5 基于糖尿病数据集实验 26

5.1 实验概述 26

5.1.1 背景 26

5.1.2数据预处理 27

5.2 实验设计 28

5.3 实验结果与分析 29

5.4 本章小结 29

6 总结和展望 29

6.1 工作总结 29

6.2 心得体会 29

6.3 进一步的工作 29

参考文献 30

致 谢 31

附录1 源代码（模块代码主体部分） 32

# 绪论

## 1.1 研究背景

随着互联网的高速发展，人们对互联网的日益依赖，全世界所产生的数据量也以惊人的量级增长。据统计，仅仅是2013年，百度、阿里巴巴以及腾讯的存储数据量都达到了数百PB（1PB = 1024TB）的级别。同时，随着国防、医疗、教育、工农业等国家计划的信息化，未来的数据只会越来越多。并且，增长的数据量意味着数据处理需要更高的计算能力，越来越多的计算任务也对计算的性能提出了更高的要求，分布式计算应运而生。分布式计算是随着大数据兴起的，在分布式计算中，重点之一便是任务的分配问题。分配问题是一种NP难度问题，有很多研究工作者提出了各种智能分配方法，极具参考意义。

2008年11月1日，中本聪（化名）在网络上发表了“白皮书”（全名“比特币——一种点对点的电子现金系统”）。加入到比特币网络中的用户节点被称为“矿工”，“矿工”通过“挖矿”获得比特币，其中完成工作量证明（pow）的节点获得比特币奖励并且可以将新区块接入到区块链上去。伴随着比特币的概念被提出至今，区块链作为比特币的底层技术，受到人们的极度重视。区块链，本质上是一个去中心化分布式的账本。近年来，随着区块链技术的蓬勃发展，人们将区块链技术应用在金融、物联网等领域中。在2013年末，以太坊（Ethereum）作为“去中心化平台”的概念被提出，以太坊提供了一个可编程区块链的平台，开发者通过编写智能合约添加到区块链上，可以利用区块链搭建分布式的应用。官方推荐的编写智能合约的语言solidity是一种图灵完备的语言，因此在可计算理论中，以太坊可以解决所有的可计算问题。

利用区块链技术的数据真实透明与共享、多方共识机制、多方安全验证等分布式处理模式用来解决机器学习的问题已经成为了一个新的研究领域。在区块链中，每一个用户可以作为一个Agent，通过将区块链视作一个多Agent组成的可交互的松散的系统，我们可以利用其分布式、并行、可复用易扩展的特点。各个Agent节点从事简单的工作，通过分布式账本对每个Agent进行调用分配，便可以实现大型的计算任务。在前人研究的基础上，本文对在区块链环境中的智能分配算力汇聚方法进行研究，具有一定的创新性和挑战性。

## 1.2 国内外研究现状

区块链本质是一种分布式计算模式，分为私有链、联盟链和公有链三种形式。对于区块链的研究热潮是近年来才兴起的新兴领域，其易于结合各行各业、公开透明、分布式的特性被众多研究者所追捧。分布式计算一直是国内外的研究热点，随着摩尔定律遇到了瓶颈，一味的缩小体积遇到了物理学上的限制。为了摆脱这种约束，采用大规模分布式的计算成为了处理大数据的有效模式。

下面对分布式计算的一些重要理论，以及基于区块链的一些应用作出简要介绍。

**1.2.1 FLP不可能性**

在物理学上存在海森堡提出的“测不准原理”，而在分布式计算领域，同样存在“FLP不可能性原理”。该原理指的是，在一个异步通信场景中，对于任意一个节点的失败，也没有任何算法能保证非失败节点达到一致性。该定理由Fischer, Lynch 和 Patterson三位作者联合发表，并且获得了Dijkstra奖。

该定理告诉了我们一个基本事实：不要浪费时间去设计一个能使得异步分布式系统，在任何场景下都能保证一致性的算法。尽管理论事实很悲观，万幸的是在工程上，我们可以通过其他尝试的方式尽量满足一致性，只不过需要多付出一点代价罢了。至于代价，在下一小节介绍CAP原理时将作出阐述。

**1.2.2 CAP原理**

CAP原理（Consistency，Availablity，Partition）表明了，对于分布式系统中，我们不可能同时满足一致性（Consistency）、可用性（Availablity）和分区容忍性（Partition），这三者不可兼得，如果想要强化某一个属性，则只能弱化其他属性的实现。

一致性：在分布式系统中，各个节点是否同一时刻有同样的值；

可用性：有限时间以内，是否任何非故障节点都能提供应答；

分区容忍性：由于节点之间的通信不可保障，因此系统需要对通信的时限进行容忍。

在设计系统时，我们往往需要牺牲系统对某个属性的支持，而选择牺牲哪一个特点需要根据具体的系统功能而定。

**1.2.3 ACID原则**

指的是对分布式数据库事务的一致性需求，由以下四个特性组成：

原子性（Atomicity）：对于某一事务里的所有操作来说，要么全部操作完成，要么整个事务失败；

一致性（Consistency）：数据库状态是一致的，不存在中间状态；

隔离性（Isolation）：各个操作之间相互隔离，互不影响；

持久性（Durability）：状态的改变不会失效，能够保持持久。

**1.2.4 区块链应用**

（1）金融服务

中国央行行长周小川曾表示，央行可能使用区块链技术支持数字货币，彻底改变传统货币的格局，目前已经有专门团队进行评估。

据2016年报道，加拿大央行正在基于区块链开发数字货币加拿大元（CAD币）。

2016年，中国邮储银行宣布推出基于区块链的资产托管系统，新用系统取消了重复的信用校验，大幅度提升了原系统的性能。

（2）众筹交易

作为互联网+金融的代表——众筹交易，由于其对于数据信息透明化的需求，相当适合区块链系统。目前，存在诸如京东金融、DAO等等众筹平台，这些平台也对区块链技术表现了充分兴趣。

（3）社交

BitMessage：一款基于区块链的p2p安全可靠的发信系统。

Twister：一款p2p的微型“微博”系统，由于其完全去中心化的特点，不需要受到管理员的权限限制。

## 1.3 论文主要工作

本文主要针对基于区块链的分布式分配算法，进行了以下工作研究：

1. 基于以太坊搭建私有链，使用私有链实现了分布式算力汇聚系统。
2. 实现合同网分配算法以及改进，并给出了相关证明过程。用改进后的分配算法搭建分布式系统并进行实验与分析。
3. 基于数据并行，使用已完成的系统实现支持对“糖尿病数据集”的回归分析。用实验评估改进分配算法的效果。

## 1.4 论文组织与结构

本论文由六个章节组成，文章的主要内容及作用分别如下：

第一章：绪论。对课题的研究背景、国内外研究现状、该论文的主要工作做了阐述，还对论文的组织与结构做出了描述。

第二章：算法基础。对本文涉及到的一些理论、概念、技术做出了简要介绍。为第三章进行算法改进提供理论基础。

第三章：算法解决方案。比较评价了当前已有的4个解决方案，改进了合同网算法，提供了数学建模并给出相应的伪代码以及流程图。

第四章：改进合同网模型实验验证。测试了本文提供的算法，对系统处于信任和非信任环境进行了实验，分析与评价了实验结果。

第五章：利用数据并行，通过糖尿病患者数据集，使用第五章实现的系统对糖尿病患者进行建模。对比不同方案的结果，并且对实验结果进行分析与评价。

第六章：总结和展望。对该课题的研究工作进行了总结，对自己的心得体会的记录，以及项目的未来的进一步工作。

# 算法基础简介

## 2.1 Agent技术

Agent是面向对象技术的抽象，是一种处于特定环境下的计算机系统。在基于网络的分布式计算中，该技术发挥着越来越重要的作用，因此得到了高度重视，甚至被一些文献视为软件领域的下一个重大突破。与对象相似的是，它封装了属性和方法，同时还具有决策能力，有良好的灵活性和自洽性。Agent技术为解决分布式计算系统提供了一个有效的概念模型，利用该技术我们可以实现多种分配方法。通常认为Agent包含有以下四个特性（目前学术界并没有准确定义）：

1. 自主性。Agent在没有外界环境干涉的情况下，能够根据自身内部状态和外界环境进行合适的行为。
2. 交互性。Agent能够与其他Agent进行交互，类似于现实世界，形成一个个组织以及团体进行活动。
3. 主动性。Agent能够主动给其他对象发送信息，一个Agent可以主动请求对方，实现自己的目的。
4. 反应性。像其他生命体一般，Agent可以对外界环境做出相应的应激行为。对于用户的操作，Agent可以做出相应的反应。

Agent技术是如此的模拟现实环境，因此对于区块链来说，每一个节点就相似于一个Agent，基于Agent模型我们可以实现良好的分布式智能分配算法。

## 2.2 分布式机器学习技术

## 2.2.1 机器学习关键技术

机器学习属于人工智能的一个分支，机器学习包含了多个学科的内容，如统计学、概率论、社会学等等，是一门多领域多学科的交叉学科。在历史上，人工智能的发展可谓是一波三折。对于机器学习算法来说， 我们一般通过建立一个模型，得到目标函数，再通过对目标函数求最大（小）值来训练参数。

在模型训练过程中，我们往往建立一个代价函数（cost function）来衡量该模型对真实世界的误差大小。因此，原问题则转化为了一个求代价函数最值的问题。在求多元函数最值问题上，存在多种方法，比如随机选取起始点的“梯度下降法”、可以概率性的跳出局部最优解的“模拟退火算法”等等。在机器学习中，我们常常用到的是随机梯度下降法，下面将简要介绍随机梯度下降法。

随机梯度下降是机器学习中一种常用的求函数最值的方法，与批处理梯度下降不同的是，随机梯度下降关注于训练数据集中的每个样本，而不是全体样本。每次迭代，该算法便更新待定参数，直到训练完毕。梯度下降的思想：

1. 对目标函数各个参数求得梯度。
2. 将参数向量根据梯度进行更新，得到新参数。
3. 将新参数带入代价函数进行计算，若满足给定误差，则退出，否则重复步骤1。

梯度下降的迭代过程数学表示如下：



其中，为需要求得的参数向量，为学习率，为参数向量的梯度。梯度下降法的缺点在于容易陷入局部最优解，尽管如此，梯度下降法也极其常用，被广泛运用于机器学习的各个领域。

## 2.2.2 logistics回归

logistics回归是一种广义的线性回归模型，主要用于疾病预测、经济预测等领域。它与多元线性回归模型很相似，而logistics回归主要处理二分类方法。首先，我们需要定义sigmoid函数，用于通过特征描述结果，往往我们选择函数



其中，。观察函数f图像易知，在数轴上f呈s型，在数学意义上在于把样本特征x带入f，将得到结果接近于0或1，这样我们就可以得到预测值。为了训练出参数矩阵，我们需要定义代价函数，然后通过代价函数求得最值，即得到矩阵，这就是我们得到的训练参数。

## 2.2.3 数据并行和模型并行

在分布式机器学习算法中，存在两种并行方式：数据并行和模型并行。

1. 模型并行

对于神经网络来说，训练过程中往往具有高度的非线性。如最简单的BP神经网络，每次迭代过程可以简单表示为：

1. 在前向传播中将训练参数输入网络并获得激励响应。
2. 在反向传播中将响应和目标输出求差，得到响应误差。

对于此类神经网络，由于存在神经之间的紧密关系与迭代，往往通过模型并行来支撑大规模的训练，模型并行将各个机器作为节点，负责处理不同层次的问题。

（2） 数据并行

本文主要采用数据并行的方式支撑机器学习的计算，同时这也是大多数场景的并行方式选择。数据并行将数据分配到不同的机器，然后将计算结果通过某种方式进行合并。数据并行由于需要组合不同的机器的训练结果，被分为了各种方法：

1. 参数平均法。将各个机器得到的训练结果取平均值，当做全局的参数值。
2. 异步更新法。与参数平均法不同的是，在工作节点与全局服务器之间，只传递更新的信息，如梯度。

基于数据并行，我们可以实现异步随机梯度下降，对于求大规模数据的梯度下降算法有极大的意义。

## 本章小结

本章对该文所涉及到的一些理论、概念、技术知识进行了一个简要介绍。

Agent技术是本文的主要思想，通过将每个节点视为一个Agent，我们能够更好的描述节点的行为。比如，描述下文算法解决方案时，我们便将参与计算的节点视为Agent，用来描述分布式智能分配算法。

为了实现分布式计算支撑机器学习技术，我们不得不对分布式机器学习技术进行一个讨论。目前学术界主要将其分为模型并行与数据并行两个模式，在本文中，我们主要使用数据并行的方式来分析评价我们的模型。

# 算法解决方案

## 相关技术

目前学术界存在很多分布式分配算法与模型，比如合同网模型、黑板模型、paxos算法、raft算法、基于马尔科夫链的决策模型、基于随机博弈的任务分配方法等。这些方法各有各的特点，由于篇幅有限，本小节对几个应用广泛的模型做一个简要的介绍与评价。

## 3.1.1 合同网模型

合同网协议（contract NET protocol，CNP）是分布式人工智能领域中的一种经典协调协议，它为人与资源、Agent之间的协调发展提供了一个良好有效的相处方式。合同网模型拥有管理者（Manager）和投标者（Contractor）两种属性的Agent，模拟了经济学中的招投标过程，通过管理者审核投标者来选择任务的发放。

**分配过程**

合同网模型可以分为4个阶段

1. 招标阶段：管理者将需要指派的任务进行发布给投标者。
2. 投标阶段：投标者收到管理者发布的任务后，参与投标，和其他投标者一起竞争该任务的归属。
3. 发放合同阶段：管理者审核投标者的标书，选择它认为最适合的投标者发放任务。
4. 签约阶段：被选中的投标者收到管理者发放的任务，和管理者签约任务的执行。

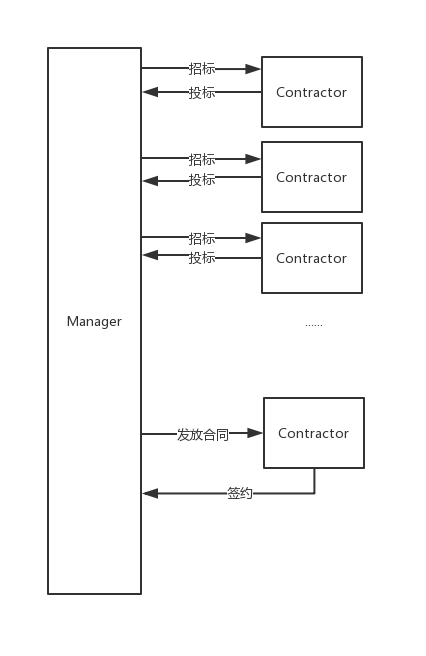
Agent之间对单个任务交互协商如图3.1所示。

图3.1 合同网模型任务交互协商图

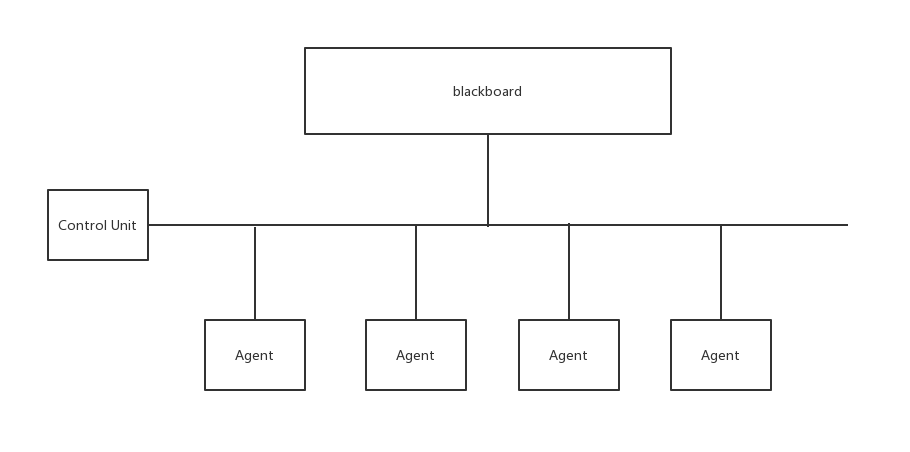
合同网模型应用广泛，其结构流程简单而又有效，受到了广泛的追捧。

## 黑板模型

黑板模型是一种协作模型，一般不会对架构产生大的影响，但对于消息结构有一个清晰的要求。在黑板模型中，黑板是一个共享问题的求解空间，Agent之间共享一块黑板。当某一个Agent解决了自己的问题，便将结果公布在黑板上，公布的信息可能会支持其他Agent求解其他任务。黑板模型包含以下几个模块：

1. 黑板：黑板可以视作一块共有的数据缓存带，所有任务信息以及Agent的处理结果都会按层次被记录在黑板上。黑板是任务的公告板，也是一个生产者，为消费者（Agent）提供待处理的项目；
2. 控制单元：控制单元负责黑板与Agent之间的交互，将生产者（黑板）和消费者（Agent）之间的耦合度降到最低。当Agent在控制单元注册之后，才能够调用控制单元的方法。控制单元直接和黑板交互，类似于中间层的存在；
3. Agent：Agent是计算的主体，控制单元和黑板都是为Agent服务而存在。Agent与控制单元交互来获取任务与提供计算结果。

黑板模型结构如图3.2所示。

图3.2 黑板模型结构图

黑板模型极其常见，其思想对于多个节点如何协作起了指导作用。

## Paxos算法

Paxos是一种很复杂的算法，它是于1990年提出的一种基于消息传递并且具有高度容错性的一致性算法。为了便于理解它的高度容错性的特性，我们使用一个形象的例子来理解：假设有一家人打算假期去游玩，但是目的地并没有确定，于是最简单的解决方案便是在微信群里投票，按照少数服从多数原则决定目的地。这是一种类似于“黑板模型”的共享内存的场景，然而Paxos考虑的复杂得多，为了实现高容错性，该算法假设如果微信服务器发生故障该怎么办？如果一家人中有一半人暂时无法取得联系，又该如何得出一致性的结果？

Paxos基于**两阶段提交协议**扩展而来，算法将节点分为了三种类型：

提议者（Proposer）：提出议案；

决策者（Acceptor）：对提议者提出的议案进行投票；

最终决策学习者（Learner）：不参与投票，只为结果做出统一性处理。

对于**两阶段提交协议**而言，整个过程可以简单的分为两步：

1. 准备阶段：提案者将自己的提案发送给决策者，观察决策者的态度，准备进入第二阶段。而决策者保留收到过的最晚的提案和接受的最大提案；
2. 提交阶段：对于某个提案者而言，提案者如果收到大部分的回复，说明它具有一定的话语权，则可以提交它的议案。然而若最终决策者得到的提案不是它之前发过的提案，它则转头支持其他提案。对于决策者而言，如果发现提案大于已经接收过的提案，则接收之并更新，反则弃置之。

最终，算法一定能得到一个最终提案，该最终提案则成为了所有节点最终的决定。然而该算法最初的论文便以复杂难懂闻名，且难以实现。根据CAP（Consistency, Availability, Partition tolerance）原理，Paxos为了保持高度的一致性牺牲了可用性，这是我们不愿意接受的。

## Raft算法

Raft算法是Paxos的一种简单实现，它将节点分为了三种类型：

Leader：处理各个节点之间的交互，当Leader不可用时，需要选出新的Leader；

Candidate：候选人，当Leader失效时，拥有被选为新的Leader的机会；

Follower：选民，负责投票选出Leader。

在Raft算法中，任何节点都有机会成为Leader。在一届任期内，整个算法分为两个阶段，一是选举过程，二是选举出来的Leader带领其他节点进行正常的操作。Raft算法有很多实现，与Paxos算法比起来更易理解。

## 总结

对于本文讨论的场景而言，其环境远没有paxos设想的那么复杂，考虑到简单易于实现的原则，本文选择合同网作为基础理论，实现适合的分配算法。合同网本身是一个简单的模型，对于现实中存在的很多问题并没有得到很好的解决（如：成本与时间的分析、恶意Agent问题），为了更好的将合同网运用到实际情况，本文对合同网模型进行了改进与优化，下一小节将主要讨论合同网改进模型。

## 改进合同网模型

## 3.2.1 改进思想

传统的合同网模型仅仅给出了分配问题的框架，然而，对于成本与时间的分析、招投标过程中Agent之间通信的损耗、恶意Agent参与竞争的问题并没有得到很好的解决。对于区块链而言，“矿工”参与运算可以得到虚拟货币的奖励，并且对于区块链上的某个节点来说，它可以与其他节点进行交易，同时，可能会有恶意矿工伪造投标消息参与竞争。

因此，为了更好的适应区块链的生态环境，本小节对合同网算法进行了本地化的改造，通过引入奖励机制“虚拟货币”、“信任度”的思想提出了新的模型。

1. 虚拟货币

对agent节点参与分布式计算的奖励，我们按任务复杂度分配给不同任务不同的奖励。事实上，我们会将高奖励的任务分配给计算能力强同时信任度高的节点。首先，为了解决问题我们需要衡量一个节点处理任务的能力。

不妨设评价函数

  （3.1）

其中， 为信任度函数（见公式3.3），表示系统对于节点的信任程度。 代表节点执行任务需要的时间。对于，通过节点执行一小段测试任务来得到。为比例系数。

对于价值越高的任务，我们优先选择评价好的节点（若该节点处于空闲状态），选择策略的映射关系数学描述为：

 （3.2）

式中，表价值函数，用来衡量任务的价值，函数值由任务发布者自己设定，对于简单任务而言往往各任务价值相等。

1. 信任度

我们无法假定所有参与节点均“诚实可靠”。有可能存在“不诚实”节点，为了投标得到更多的任务，有心伪造自己的算力情况，欺骗管理者。或者因为其他不可抗性原因导致算力提交出现了错误。自私节点以自己利益最大化为目的，如果没有考虑到自私节点的情况，系统会将大量任务分配给自私节点，而由于自私节点的算力不足无法迅速得到结果，导致系统进入漫长的等待，严重拖慢了整个系统的运行效率。考虑到可能存在恶意节点的情况，我们引入拜占庭将军问题的描述进行类比。

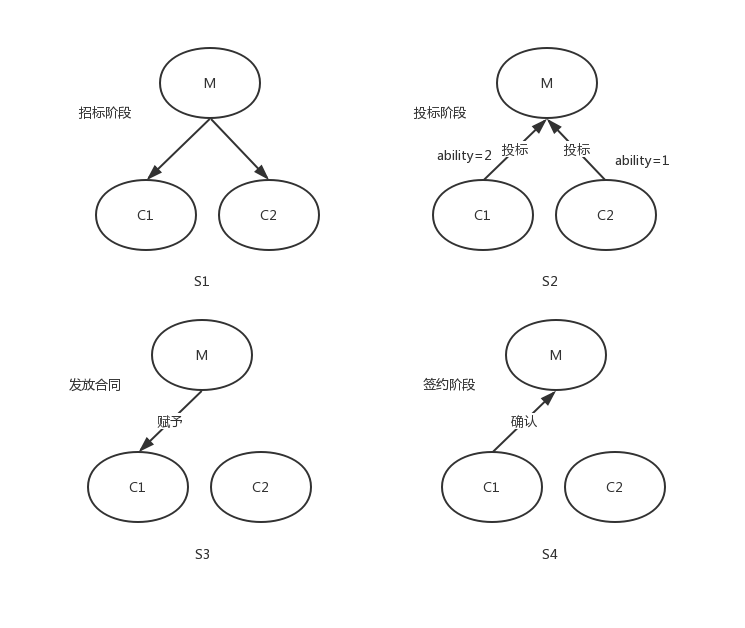


图3.3 欺骗节点行为示意图

图3.3分为4个阶段，其中M为管理者，C1为“欺骗节点”，C2为诚实节点。由于C1伪造了自己的能力，使得管理者相信C1比C2更具备解决任务的能力，于是将合同分配给了C1。为了解决这个情况，我们引入“信任度”的概念，用来衡量一个节点的可信程度。类比于“芝麻信用”的思想，为了描述可信程度，需要将该节点过去的行为进行一个评估，用来确定它的信用值。

设表示节点的信用值，有以下数学描述：

 （3.3）

从公式可以看出，若节点未有过“欺骗”情况，则信用度为1，同时“欺骗”的情况越多，节点信用度越低，将严重影响节点的能力评估。

## 3.2.2 数学建模

首先，对分配关系数学建模如下：

**Step1 参数描述**

定义二元组 ，其中集合 表示所有参与计算的节点Agent的集合，集合 表示任务集合。

不妨设有m个Agent，同时存在n个任务，则分配矩阵定义如下：

 （3.4）

定义代价矩阵 ，表示将任务 分配给 的代价。

**Step2模型建立**

**定义代价函数**

  （3.5）

表示节点执行任务所需要的代价。其中 代表节点执行任务需要的时间。 表示节点执行任务的当前负载。 为信任度函数（见公式3.3），表示系统对于节点的信任程度。 与 与 均为比例系数。

**定义支出函数**

 （3.6）

3.6式中， 表示执行任务需要的支出。 表示系统为任务的估价函数，通过虚拟货币的形式发放。 为比例系数，根据环境变化。

根据方程式（3.5）（3.6），易得

 （3.7）

对于整个模型来说，我们需要一个效能指标函数，来衡量分配方式的好坏。根据分配矩阵与代价矩阵，由（3.4）（3.5）（3.6）（3.7）式可以很容易得到

 （3.8）

3.8式中，Sum是关于的函数，表示在整个分配过程中，分布式集群执行任务的好坏程度。更一般的，由于同一任务往往只分配一次，观察分配矩阵 的结构，由（3.8）易得

 （3.9）

我们可以得到一个最优解 ，表示全局分配最优的情况，此时

 （3.10）

**Step3约束条件**

约束条件需满足如下情况：

 （3.11）

（3.11）式表示将所有任务分配完毕。

 （3.12）

（3.12）式表示若任务与有顺序关系，则需要让他们顺序执行。

 （3.13）

（3.13）式表示对于任务的定价需要高于任务的花费，让Agent有充分意愿执行。

**Step4小结**

该模型是对合同网算法的本地化处理，为下一节的详细设计中算法改进过程提供了支撑。分配问题是一个NP难题，本文无法做到对所有的分配方式进行讨论，因此该模型有一定的局限性，只是基于前人研究的改进与创新。该模型说明了，只要通过测试任务测试得到每一个节点的算力大小，对于本文选择的简单分配方式就可以得到一个表现良好的分配矩阵。

由于每次的任务的发布可能是动态添加的，因此对于系统而言，数据传输过程便成为了制约整个算法的瓶颈。在实际处理中，我们在一开始对整个系统的节点进行算力测试，构造分配矩阵，每次执行任务完毕后评估Agent执行结果并部分更新分配矩阵，后续则按照合同网招投标流程处理。

## 3.2.3 详细设计

本小节对于算法流程进行详细的描述。

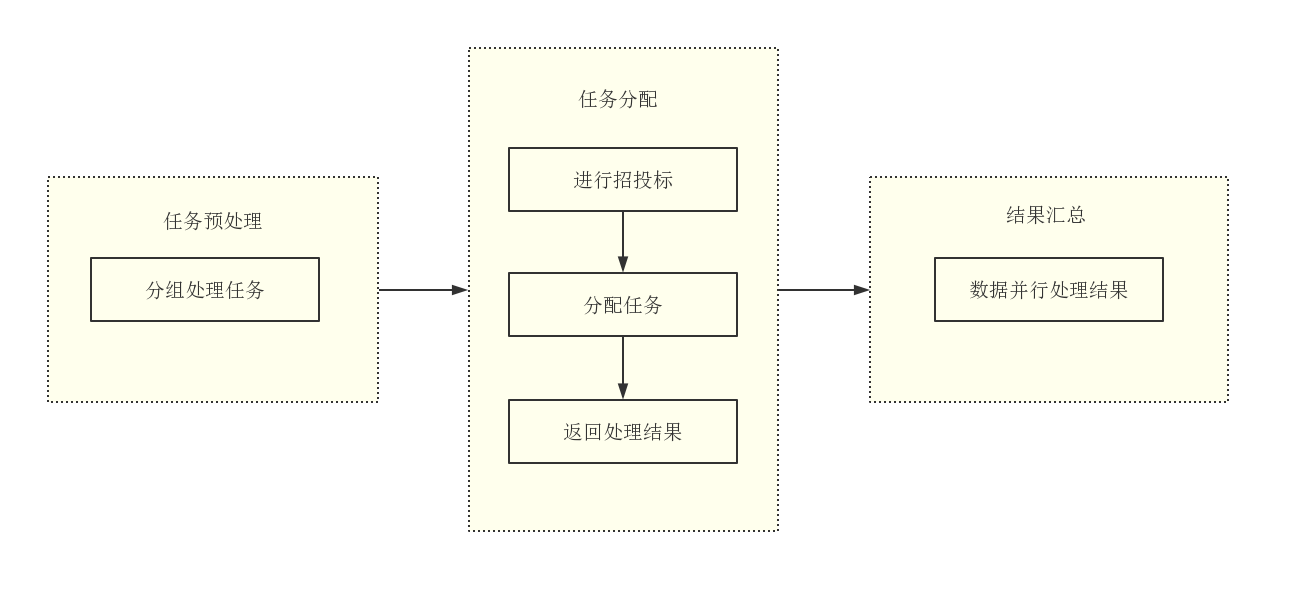
第三章对于整个分配流程有一个整体描述，第四章将对改进的算法进行实验。整个算法分为以下几个步骤

（1）任务预处理：将机器学习任务分批并编号；

（2）任务分配：把任务分配给相应的节点；

（3）结果汇总：将各个节点得到的结果汇总并分析。

其完整流程如图3.4所示

图3.4 任务分配完整流程图

当管理者（Manager）接收到一个新任务时，需要对该任务进行定价，然后发起招标请求。管理者（Manager）分析接收到的招标情况，选择将任务分配给最合适的对象，并且设置期望时间。节点接受到标书后确认中标并执行任务，运行完毕后将结果返回给管理者（Manager）。管理者对执行时间与根据节点标书得到的期望时间作对比，若发现该节点存在“欺骗”现象，则降低该节点的可信度。若任务在deadline之前依旧没有执行完毕，则重新对其他节点进行招标。图3.5展示了对于单个任务处理的UML活动图。

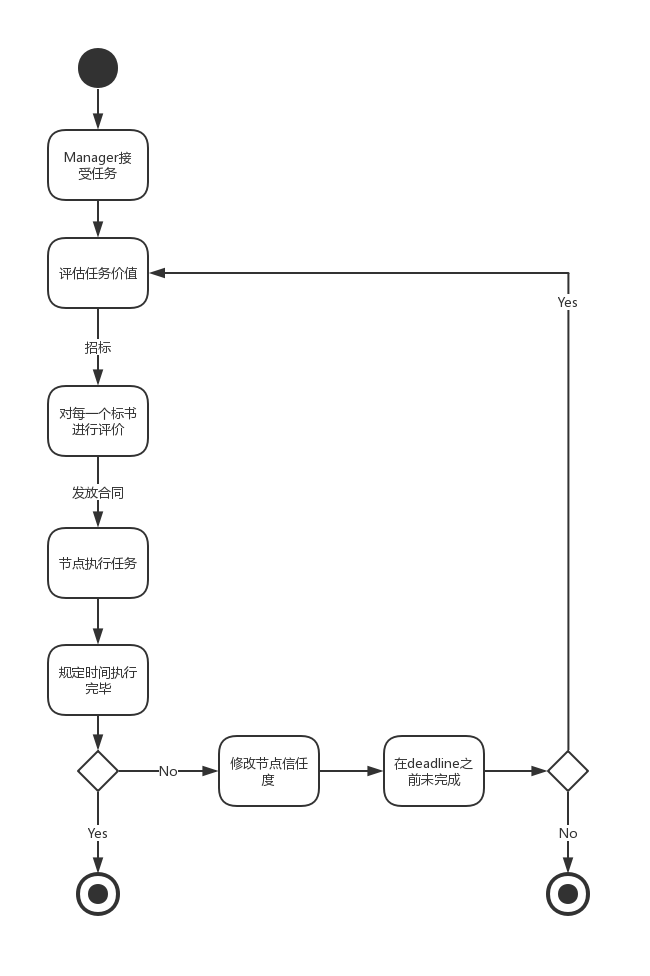


图3.5 单任务任务分配活动图

为了更清晰方便的描述分配过程，算法的伪代码流程如下所示。

|  |
| --- |
| **Algorithm 1** pseudocode for this algorithm |
| 1. **Input**: tasks; 2. **Output**: combined result; 3. **Initialization**: successTaskNum0, totalTaskNumtasks.length, nodeNumber1 4. check connected node, then **do** 5. nodeNumbernodes.length; 6. set tasks value, and evaluate for every nodes. 7. testTaskUint for every nodes, then divide tasks for lots of groups. 8. **while** successTaskNum < totalTaskNum **do** 9. **if** unSuccess.length > 0,  **then** 10. group = unSuccess.pop(); 11. **else** 12. group = groups.pop(); 13. **end if** 14. Select property nodes, and then give out invitation. 15. Listen node's response, then give out one group for selected node. 16. Check node's response. 17. **if** actualTime > estimatedTime, **then** 18. node.successNum --, unSuccess.push(group); 19. **else** 20. successTaskNum += groupCount; 21. **end if** 22. **end while** 23. Combine results for next step. |

## 本章小结

本章对目前被广泛使用的分布式算法进行了一个简要的介绍，对比评价了各个算法的优劣性，并选择合同网算法作为本次实验的基础。而对于合同网模型的缺陷，在3.2节，本章对该模型进行了本地化的改造。在3.2.2节建立了新的数学模型，在3.2.3节详细设计中给出了新模型的活动图以及伪代码，为下一章的实验提供了理论基础。

# 改进合同网模型实验验证

## 实验环境

本文基于以太坊平台搭建私有链进行实验，开发框架使用truffle平台，并通过VMWare创建虚拟机加入私有链来模拟各个节点，各个节点之间通过NAT模式与主机连接。我们通过搭建的系统实现区块链分布式系统的交互过程。系统的具体参数如下所示：

**硬件平台**

|  |  |  |
| --- | --- | --- |
|  | 主机 | 虚拟机 |
| CPU | Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz | cpu cores：1 |
| GPU | NVIDIA GeForce GTX 950M |  |
| RAM | 8GB | 1GB |

**软件平台**

|  |  |  |
| --- | --- | --- |
|  | 版本 | 描述 |
| Truffle | v5.0.4 | 以太坊开发框架 |
| Ganache | v1.1 | 以太坊测试链 |
| Solidity | v0.5.0 | 实现任务调度 |
| Node.js | v10.15.1 | 利用Mocha框架用于测试 |
| Python | v3.7 | 用于实现机器学习算法以及数据可视化 |
| JavaScript、HTML、CSS |  | 前端可视化平台搭建 |

## 实验设计

在我们实现的系统中，为了起到对照实验的效果，设置两类管理者（Manager）Agent与两类投标者（Contractor）Agent。两类管理者的区别在于对于投标的选择策略不同。其中一类管理者采用简单发放标书的策略，即对于投标者投递的标书按接收顺序给节点分配，先到先得，不考虑其他因素。第二类管理者则结合信任度的思想，通过评估函数（见公式3.1）计算该节点的价值，然后从标书中选择价值高的节点来给予任务。可以预见的是，这两类管理者适用于不同的环境，若所有节点都诚实守信，那么直观上两个算法效率相似；若系统处于非诚实环境，即存在不诚实节点，恶意谎报标书，那么第二类管理者在这种环境下会比第一类管理者工作效果好。管理者类型如表4.1所示。

|  |  |
| --- | --- |
| 管理者类型 | 分配策略 |
| 类型A | 按顺序分配 |
| 类型B | 按评估函数分配 |

表4.1 管理者类型与分配策略

按照同样的思想，接下来我们设计两类投标者（Contractor）。其中一类投标者诚实守信，对于整个系统来说它们是老实节点，从不造假，并且有任务执行时不会参与对其他任务的投标。第二类投标者是所谓的恶意节点，它们以自身利益为重，不考虑整个系统的性能。为了方便描述，对于第二类恶意节点，我们定义一个恶意随机变量。该变量的现实意义是对所有任务参与投标，且每次投标时，该节点有的几率通过设置算力的最大值从而百分百得到该任务。其中，即服从0—1分布。投标者类型如表4.2所示。

|  |  |
| --- | --- |
| 投标者类型 | 投标策略 |
| 类型A | 诚实守信 |
| 类型B | 行为受到恶意随机变量控制 |

表4.2 投标者类型与投标策略

为了方便起见，我们创建4个节点，同时设置单元任务为对10000个随机数的快速排序，通过设置不同的单元任务数量提供给不同类型的Agent执行，我们便可以得到实验数据。我们进行两组实验，分别测试两种类型的管理者在两种环境下的表现。实验一为类型A与类型B的管理者在完全诚实合作下进行协作任务，实验二则为两类管理者在非诚实环境下协作任务。每一组实验类型A与类型B管理者互为对照组。

## 实验结果与分析

## 4.3.1 实验过程数据

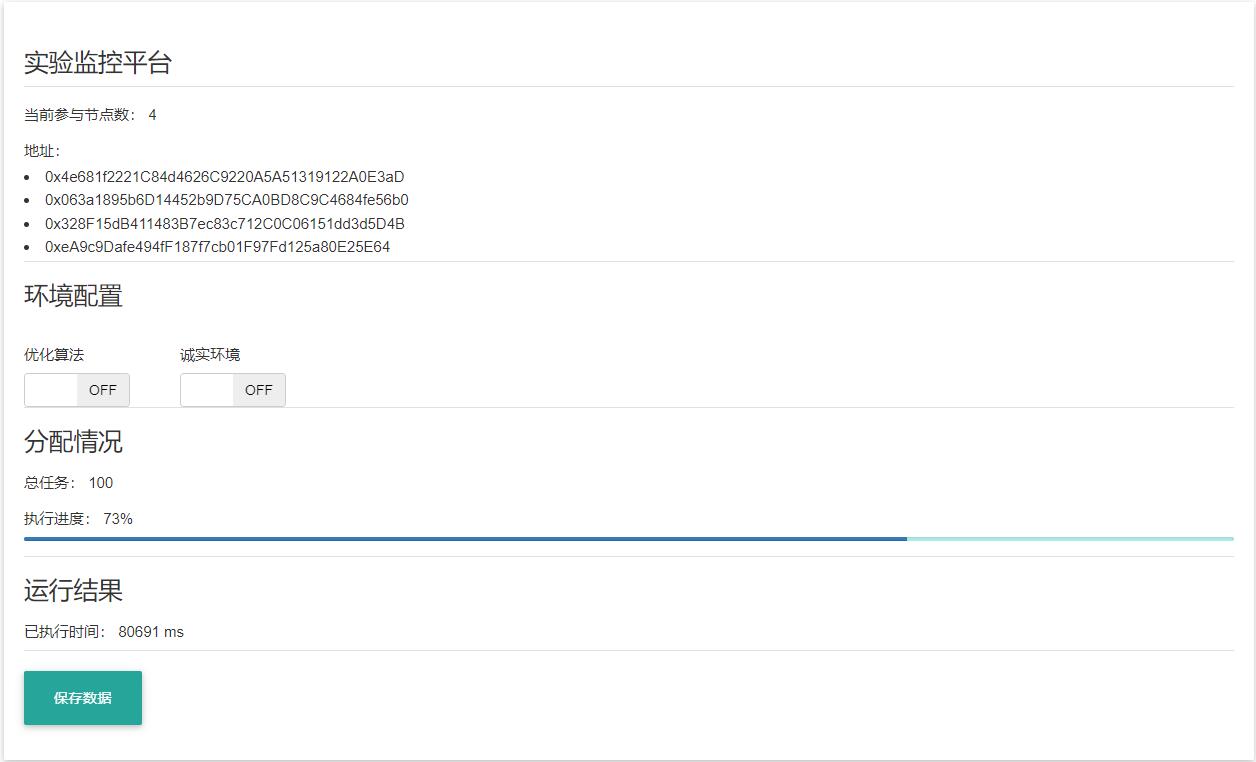
在实验过程中会产生很多中间数据，为了更好的观察实验结论，我们利用bootstrap框架搭建了一个简单的前端平台，用来展示运行算法过程中产生的数据。该平台会间隔一定的时间读取区块链上的数据，以达到监控的效果。取某一次实验中区块上记载的实验数据快照如图4.1所示

图4.1 实验快照

从图中可以看出，该平台的结构分为5个部分：

第一部分是参与节点信息，表示了参与节点的数量及地址；

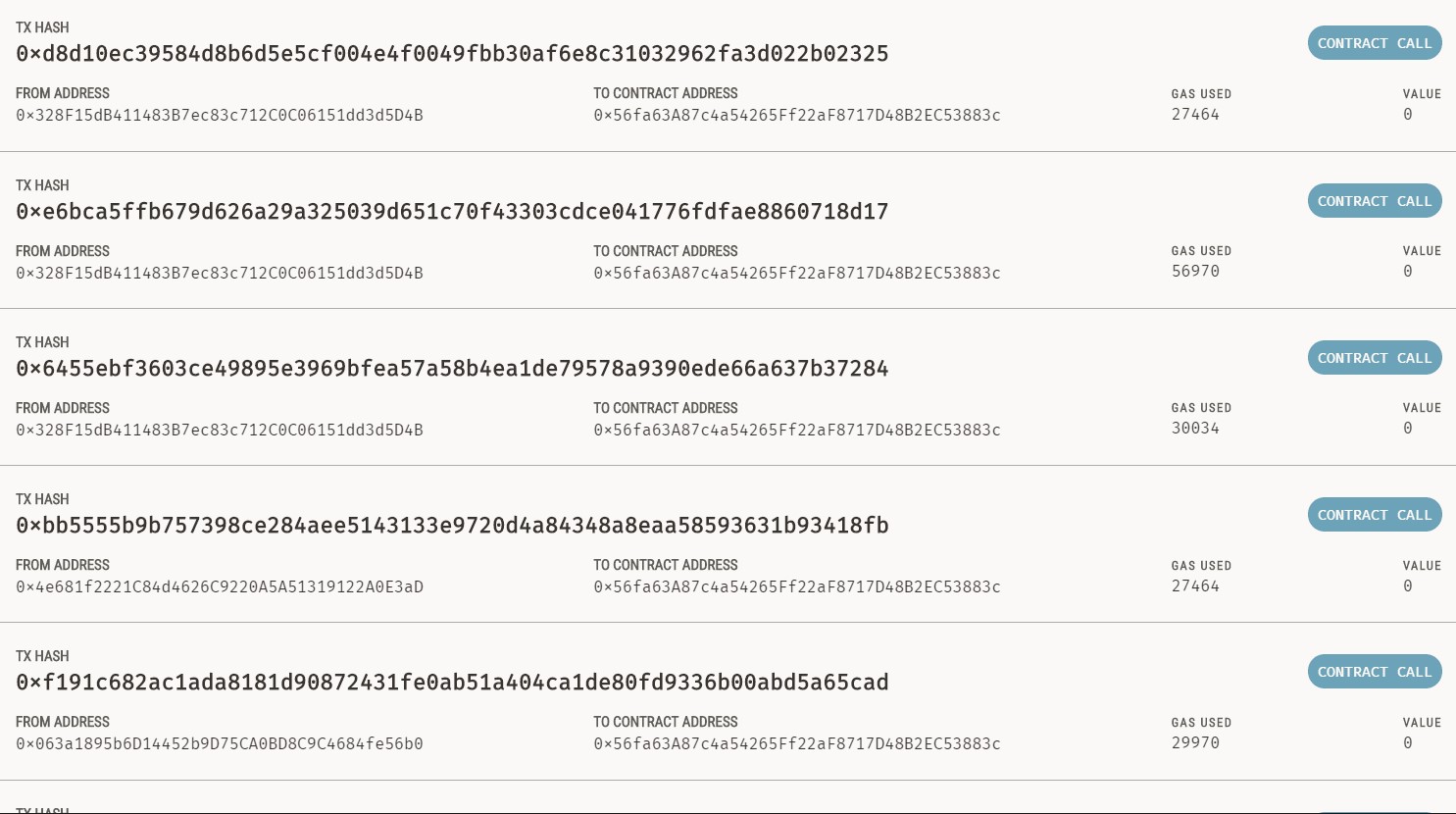
第二部分是关于实验环境配置信息，可以选择是否启用优化算法，以及当前是否为诚实环境；

第三部分展示了任务完成情况，进度条显示了当前执行进度；

第四部分为已执行时间；

第五部分可以保存当前执行状态数据。

同时，节点每次招投标都通过一次交易来完成，账户产生的每一笔交易都记录在本地客户端中，如图4.2所示

图4.2 交易记录

这些交易记录保存了每次招投标的状态与结果。

在算法运行过程中私有链会生成新的区块，生成的区块记录如图4.3所示

图4.3 区块记录

## 4.3.2 实验结果分析

**实验一（完全诚实环境）**

首先，设定环境为完全诚实合作的，即所有的投标者都是类型A，我们来比较不同类型的管理者对于结果的影响（图4.4）。该组实验共有4个诚实节点加入运算，实验数据为4次重复实验取的平均值（表4.1）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 任务数 | 50 | 100 | 150 | 200 |
| 管理者类型A/ms | 55094 | 104914 | 158476 | 203867 |
| 管理者类型B/ms | 54932 | 99941 | 156932 | 201743 |

表4.1 实验一

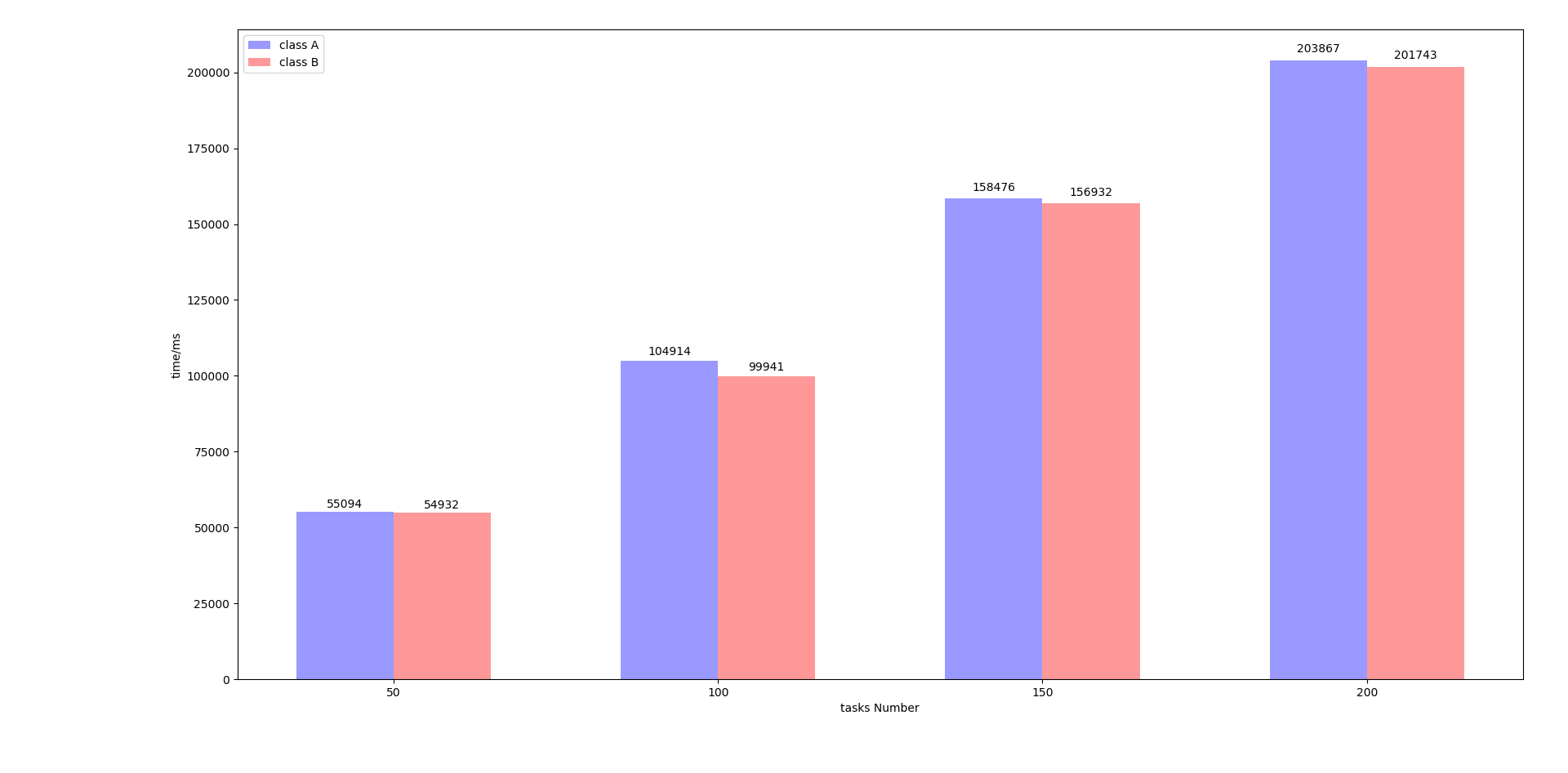


图4.4 实验一

在实验一中，我们分别取任务数量为50、100、150、200，类型A与类型B的数据如图4.4所示。从图表可以看出，在环境为完全诚实合作时，类型B管理者与类型A管理者表现相差无几，但类型B表现稍微好一些。直觉上分析，对于类型A来说，由于按顺序分配标书，会导致有充分算力的节点不能得到充分利用。对于类型B来说，由于利用了评估函数（式3.1），测试得到每个节点的能力，可以充分发挥节点的算力水平。

**实验二（非诚实环境）**

下面进行实验二，测试两种类型管理者在非诚实环境下的表现。我们来比较不同类型的管理者对于结果的影响（图4.5）。该实验共有4个节点参与计算，其中有两个诚实节点，两个恶意节点，实验数据为4次重复实验取的平均值（表4.2）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 任务数 | 50 | 100 | 150 | 200 |
| 管理者类型A/ms | 56726 | 117421 | 173741 | 236907 |
| 管理者类型B/ms | 55412 | 114597 | 163948 | 220110 |

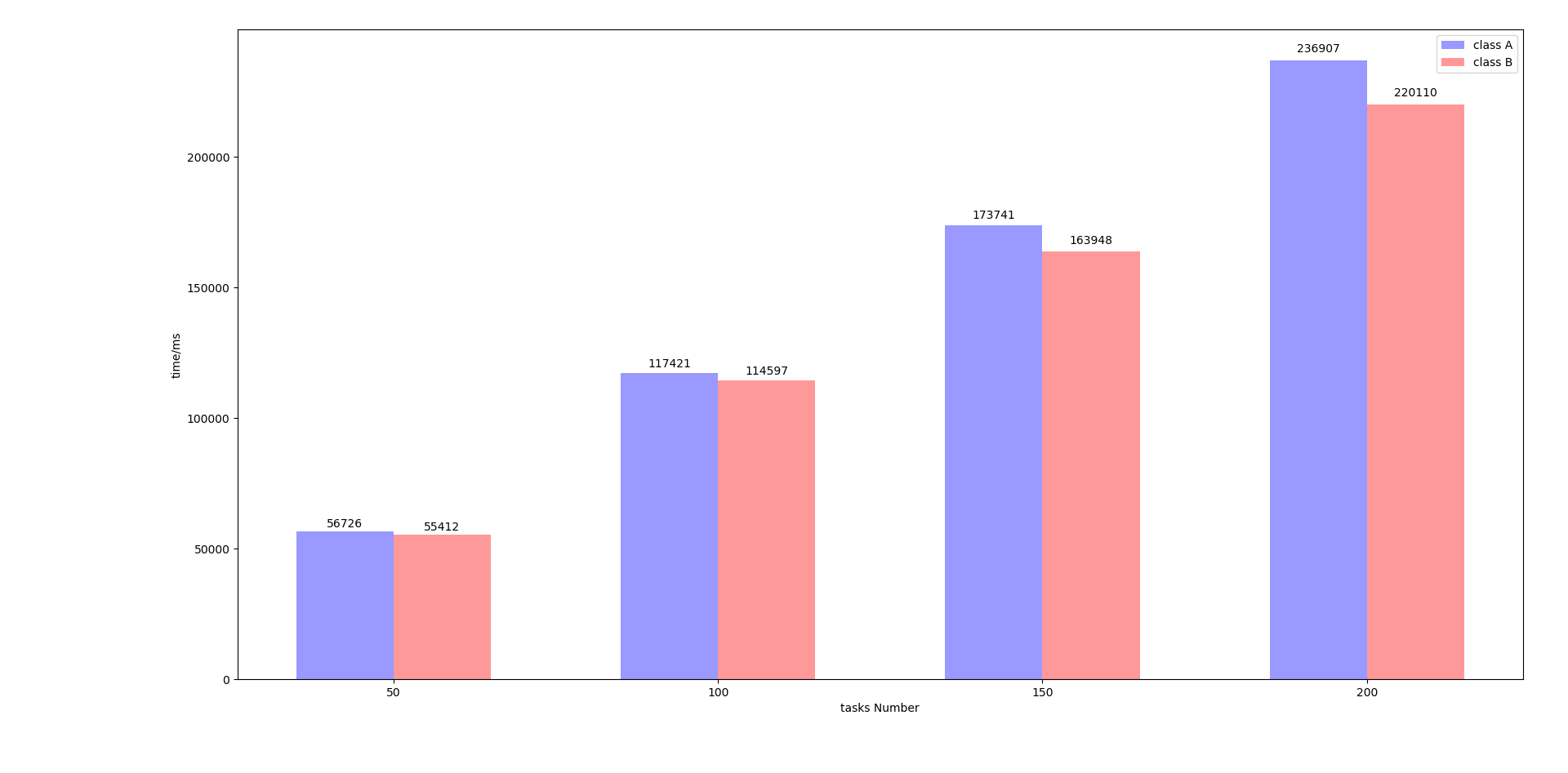
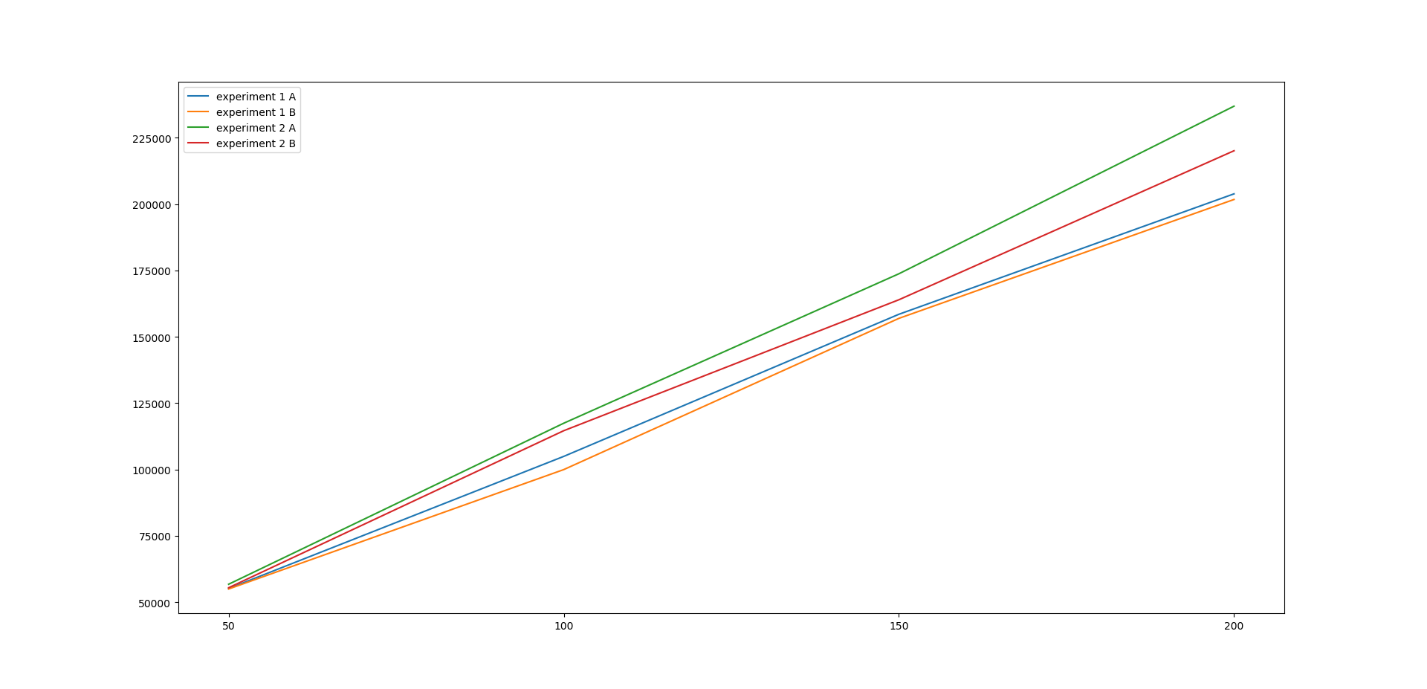
表4.2 实验二

图4.5 实验二

从图4.5可以看出，在非诚实环境下，优化算法的表现明显比原始算法效果好，并且随着任务数的增多，效果会越来越好。对比实验一与实验二的图表，我们可以发现在不完全信任的环境中系统处理任务时间成本明显上升。我们对实验结果进行分析，当存在有欺骗节点时，由于欺骗节点伪造了自己的算力，使得管理者会将任务分配给欺骗者，其他正常节点得不到任务，导致浪费与管理者交流的机会，时间成本直线上升。在优化算法中，我们由评估函数（式3.1）对节点的算力进行了调整，并且随着任务数的增加，根据大数定律，评估结果会越来越准确，最终欺骗节点与正常节点之间的算力之比会调整到一个正常值。

实验一与实验二折线图如图4.6所示。

图4.6 实验一和实验二趋势图



## 本章小结

从第三章算法设计到第四章实验验证，我们得到了一个适用于非诚实环境的分配算法模型。该模型在完全信任的环境中表现一般，在非完全信任的环境下表现良好。在实验过程中，发现系统损耗的时间往往来自于管理者（Manager）与投标者（Contractor）进行通信时的损耗，这也是一个可以优化的点。

# 基于糖尿病数据集实验

## 实验概述

## 5.1.1 背景

本章使用前面章节搭建的分布式平台，利用logistics回归对糖尿病模型进行预测。本实验数据来自“美国国家糖尿病和消化与肾脏疾病研究所”，实验数据共有768条，内容如表5.1所示。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |

表5.1 单条糖尿病数据示例

其中，我们通过8个影响因子来判断一个人是否患有糖尿病，下面简要解释每一条数据的内容：

Pregnancies：怀孕次数；

Glucose：葡萄糖，口服葡萄糖耐量试验中血浆葡萄糖浓度；

BloodPressure：血压，舒张压；

SkinThickness：肱三头肌皮肤皱褶厚度（mm）；

Insulin：胰岛素，2小时血清胰岛素水平；

BMI：身体质量指数，体重/（身高\*身高）；

DiabetesPedigreeFunction：糖尿病系谱功能，根据家族史评估糖尿病可能性；

Age：年龄；

Outcome：结果，1表示患有糖尿病，0表示未患。

## 5.1.2数据预处理

下面我们对数据特征进行研究。

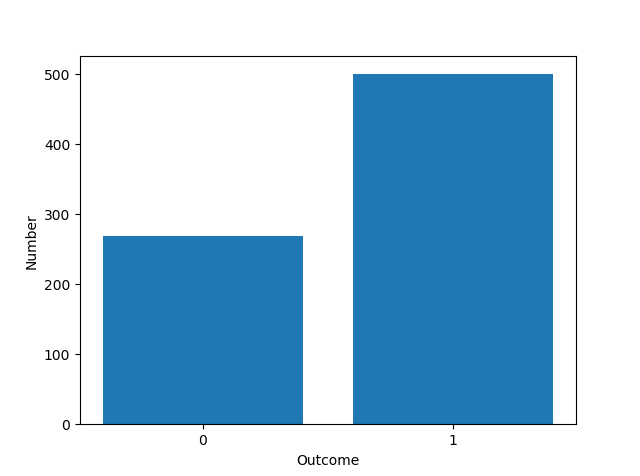
在该数据集中，糖尿病患者有500人，非患者有268人，见图5.1

图5.1 患者与非患者数量

由于在数据集中存在着大量的空数据（如血压为0），我们需要对其进行数据清洗，原则上将缺失的数据用平均值替代。同时，为了提升精度，我们将训练数据标准化，把数值特征缩放成均值为0，方差为1的状态。为了更加直观看出数据与数据之间的关系，我们需要了解各个数据之间的相关程度。计算相关性最常用的是皮尔森相关系数，用来反映两个变量的线性相关程度，公式如5.1所示

 （5.1）

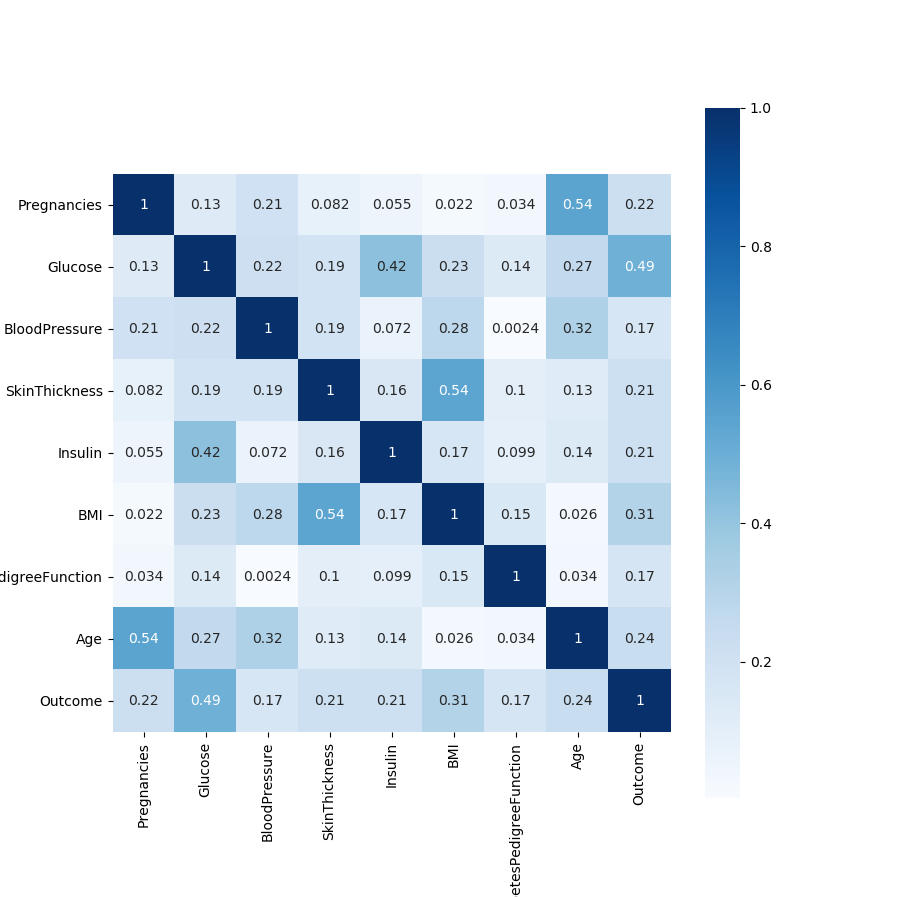
得到的相关性热力图如图5.2所示。

图5.2 相关性热力图

## 实验设计

对于训练模型，考虑到该分类问题输出值为0和1，我们选择使用logistics回归对数据进行拟合。

我们选择使用交叉验证法来进行实验。首先将数据分成两份，实验数据和测试数据共750条，剩余18条数据用于交叉检查模型。对于750条数据集，我们将其分为10个大小均为75的互斥子集，即，每个子集尽可能保持数据一致性，即从数据集中分层采样而来。然后，我们每次使用相邻9个子集的并集作为训练集，剩下的那个子集作为测试集，这样我们就得到了10组训练/测试集，从而可以进行10次训练和测试。最后，我们将10次的训练结果取平均值就得到了最终的训练模型数据。

在实验中，我们将这10次训练/测试任务作为10个任务单元，分配给节点去执行，这样我们就能得到训练结果以及执行效率情况。

## 实验结果与分析

## 本章小结

# 总结和展望

## 工作总结

## 心得体会

## 进一步的工作

# 参考文献

# 致 谢

# 附录1 源代码（模块代码主体部分）