高频交易下的股票涨跌预测

梅智敏

院 （系）： 计算学部 专 业：软件工程

学　　号： 1183710118 指导教师：张彦航

**2022年5月**



**毕业设计（论文）**

题 目 高频交易下的股票涨跌预测

专 业 软件工程

学 号 1183710118

学 生 梅智敏

指 导 教 师 张彦航

答 辩 日 期

# 摘 要

在数字经济时代，随着数据的规模越来越大，如何压缩和存储是一个基础且重要的课题。然而，传统的数据压缩算法只将重点放在如何提升压缩率，使之接近香农提出的熵的下界，这些算法不适用于全部的场景，特别是对于压缩后的数据有查询需求（如在避免解压全部数据的情况下访问部分数据）时。因此需要一种算法来适应存在着大量对压缩数据有实时查询需求的应用的系统。且随着物联网技术的不断发展、数据的不断增加，基于云的物联网解决方案渐渐无法满足企业的需求。此时边缘计算作为云的扩展，可以加快数据分析的速度，便于企业更快更好地做出决策，但边缘硬件也对算法提出了新的要求，如存储空间更小、不适合做批处理等。因此，本文在边缘硬件环境下实现了基于有损压缩、可对压缩数据进行分析式查询的Plato算法，并通过利用误差向量的正交投影性质，进一步改善了向量乘法运算的error-bound；实现了基于无损压缩、支持随机访问的GD算法，并通过改进预处理时的终止条件，避免了预估空间落入局部最优值。

关键词：时序数据；数据压缩；边缘硬件；误差度量；随机访问

# Abstract

In the era of digital economy, with the increasing scale of data. How to compress and store is a basic and important topic. However, the traditional data compression algorithms only focus on how to improve the compression rate and make is close to the lower bound of Shannon’s entropy.These algorithms are not suitable for all scenarios especially when the compressed data needs to be queried(such as accessing part of the data without decompressing all the data). Therefore, an algorithm is needed to adapt to the system with a large number of real-time query requirements for compressed data. With the continuous development of Internet of things technology and the increasing of data, cloud based Internet of things solutions are gradually unable to meet the needs of enterprises. At this time, as an extension of the cloud, edge computing can speed up the speed of data analysis, which is convenient for enterprises to make decisions faster and better. But edge hardware also puts forward new requirements for the algorithm, such as smaller storage space, not suitable for batch processing and so on. Therefore, in the edge hardware environment, this paper implements the Plato algorithm based on lossy compression, which can query compressed data analytically, and further improves the error bound of vector multiplication by using the orthogonal projection property of error vector; The GD algorithm based on lossless compression and random access is implemented. By improving the termination condition of preprocessing, the prediction space is avoided to fall into the local optimal value.

**Keywords:** time series, data compression, edge hardware, error measurement, random access

# 目 录

[摘 要 I](#_Toc73716409)

[Abstract II](#_Toc73716410)

[目 录 III](#_Toc73716411)

[第1章 绪 论 - 1 -](#_Toc73716412)

[1.1 课题背景及研究的目的和意义 - 1 -](#_Toc73716413)

[1.2 国内外研究现状 - 4 -](#_Toc73716414)

[1.2.1 概述 - 4 -](#_Toc73716415)

[1.2.2 压缩技术分类及介绍 - 4 -](#_Toc73716416)

[1.2.2 已有算法分类及概述 - 5 -](#_Toc73716417)

[1.3 该领域存在的挑战及本文的主要研究内容 - 7 -](#_Toc73716418)

[1.3.1 该领域存在的挑战 - 7 -](#_Toc73716419)

[1.3.2 主要研究内容 - 7 -](#_Toc73716420)

[1.4 章节安排 - 8 -](#_Toc73716421)

[第2章 Plato算法的实现与改进 - 10 -](#_Toc73716422)

[2.1 引言 - 10 -](#_Toc73716423)

[2.2 Plato数据压缩 - 10 -](#_Toc73716424)

[2.2.1 时序数据分段 - 10 -](#_Toc73716425)

[2.2.2 估计函数选择与存储 - 12 -](#_Toc73716426)

[2.2.3 误差度量 - 12 -](#_Toc73716427)

[2.3 Plato数据查询 - 13 -](#_Toc73716428)

[2.3.1 误差传播 - 14 -](#_Toc73716429)

[2.3.2 时序数据操作的误差界分析 - 15 -](#_Toc73716430)

[2.3.3 使用正交投影性质改进误差界 - 16 -](#_Toc73716431)

[2.3.4 段不对齐的情况下改进误差界 - 17 -](#_Toc73716432)

[2.3.5 对误差界的进一步改进 - 19 -](#_Toc73716433)

[2.4 本章小结 - 20 -](#_Toc73716434)

[第3章 GD算法的实现与改进 - 21 -](#_Toc73716435)

[3.1 引言 - 21 -](#_Toc73716436)

[3.2 GD数据压缩 - 21 -](#_Toc73716437)

[3.2.1 预处理 - 22 -](#_Toc73716438)

[3.2.2 压缩 - 23 -](#_Toc73716439)

[3.2.3 压缩文件大小分析 - 25 -](#_Toc73716440)

[3.3 GD数据解压缩与随机访问 - 25 -](#_Toc73716441)

[3.3.1 GD数据解压缩 - 26 -](#_Toc73716442)

[3.3.2 GD随机访问及代价分析 - 27 -](#_Toc73716443)

[3.4 本章小结 - 29 -](#_Toc73716444)

[第4章 实验设计与分析 - 30 -](#_Toc73716445)

[4.1 实验环境 - 30 -](#_Toc73716446)

[4.2 Plato算法分析 - 31 -](#_Toc73716447)

[4.2.1 数据集及预处理 - 31 -](#_Toc73716448)

[4.2.2 实验分析 - 31 -](#_Toc73716449)

[4.3 GD算法分析 - 35 -](#_Toc73716450)

[4.3.1 数据集及预处理 - 35 -](#_Toc73716451)

[4.3.2 实验分析 - 35 -](#_Toc73716452)

[4.4 本章小结 - 38 -](#_Toc73716453)

[结 论 - 40 -](#_Toc73716454)

[参考文献 - 41 -](#_Toc73716455)

[原创性声明 - 43 -](#_Toc73716456)

[致 谢 - 44 -](#_Toc73716457)

# 第1章 绪 论

## 1.1 课题背景及研究的目的和意义

### 1.1.1 高频交易

在过去的几十年里，新技术正在或者已经改变了股票以及其他金融产品的交易活动。在计算机出现之前，所有的股票交易都是在交易大厅里面人与人之间进行，后台其他人员的处理活动确保了交易的顺利完成。但是，后台处理和实际交易过程中已经被逐渐地自动化。而高频交易就是自动化交易的一个子集，它通常被定义为利用计算机交易算法来提交交易指令和取消交易指令。根据美国证券交易委员会的描述，高频交易具有以下的一些特征：（1）使用高性能的计算机以及复杂的计算机应用程序来产生，路由和执行交易指令。根据最新的行情信息和宏观消息计算产生交易信号是高频交易最核心的功能之一。此外，另有监控系统对交易策略进行实时监控，当异常事件发生时便实时发出警告。（2）将服务器直接架设在交易所，最小化由网络等原因造成的延时在极短的时间内创建和实施买卖交易。（3）采用逐笔数据进行计算。高频交易往往需要以毫秒级的逐笔数据作为计算基础，以消化最新的市场信息，并以此制定交易决策。与人工交易比起来，程序化交易能够更快、有效、理性地决策，这就使得程序化成为高频交易不可或缺的一部分。（4）买卖频繁，持仓时间短。

### 1.1.2 限价订单簿Limit Order Book

传统的金融市场是以报价驱动为基础而运作的，即交易员与交易商（也称“做市商”）在市场中进行沟通。交易商负责维护金融资产的库存量，公布出价和要价，并被要求按其报价进行交易。交易商的存在提供了市场流动性，但是报价驱动市场的透明度相对较低。

而如今，随着电子通信网络和高频交易的快速发展，订单驱动的金融市场成为了主流。它和传统的报价驱动市场最大的区别就在于市场中的交易员可以直接与其他交易员进行交易而没有中间交易商的干预。而订单驱动的金融市场的核心就是以限价订单簿Limit Order Book为中心的双重拍卖机制来促进交易，这一机制已经被全球各大交易所所广泛采用，例如纽约证券交易所（NYSE）、纳斯达克和伦敦证券交易所（LSE）。

在以Limit Order Book为中心的金融市场中，交易者可以提交两大类交易指令：限价指令Limit Order和市场价指令Market Order。前者会要求该交易的成交价格必须好于或者等于所指定的值，如果当前市场中最好的价格仍然不能满足要求，这个交易指令就会继续等待直到市场中出现了满足要求的价格为止，所以此类指令具有“执行不确定性”，即他有可能不会被执行，一直在等待状态。后者则要求该交易立刻以当前市场中最好的价格来完成订单，此类订单一定会被执行，但是不能保证成交价格足够好，所以它具有“价格不确定性”。另外，除了提交这两类指令，交易者也可以取消他们之前所提交但是暂时处于等待状态的Limit Order。

本质上来说，Limit Order Book（也称为 LOB）是当前金融市场行情的一个快照，其基本功能是匹配市场中的买卖双方，它由股票交易所统一管理并呈现给每一位交易者，以帮助交易者们做出交易决策。LOB包含了一组可以代表市场状态及其变化方式的数据且会动态更新，它是当今大多数电子交易所在金融和加密货币市场中使用的机制。图1是LOB的示意图，可以帮助我们直观了解。

**独立统计：**每一支股票的LOB数据分开统计，每一个LOB只反映其对应的那支股票的信息。

**买卖两方**：Ask代表卖方，Bid代表买方。

**价格数量对应关系**：它表明当前市场中有人愿意以多少价格出售/买入多少数量的股票，以Ask方的20.6价格为例，它对应的Volume值为2，代表目前市场中有人愿意以“不低于20.6”的价格出售2支股票。需要注意的是，展示在Limit Order Book中的价格都是Limit Order中所要求的价格，而Limit Order是给出某个价格底线，然后要求成交价“好于或者等于该底线价格”。故Bid方20.5价格对应Volume值为3的含义就是：当前市场中有人愿意以“不高于20.5”的价格买入3支股票。

**买卖双方不同的Level：**仅仅依据交易者的预期价格而无需考虑对应的Volume来对他们提交的“交易需求”进行分级，Level-1是最高级，当市场中的交易发生时，会优先匹配高等级的“交易需求”再到低等级的需求。首先对于Bid方，价格越高则所处的等级越高，故20.5的价格处于Level-1，20.4的价格处于Level-2。接着是Ask方，价格越低则所处的等级越高，20.6的价格处于Level1，20.7的价格处于Level2。

**LOB的更新：**每当市场中有交易员提交了新交易指令或者取消了之前的某个Limit Order，LOB都会更新以呈现出当前市场的真实交易情况给每一位交易者。图2和图3详细展示了两种情况下的LOB更新过程：（1）当有交易者提交了新的交易指令时，LOB会发生更新，图2中展示的是限价为20.55的Limit Ask Order取代20.6成为了新的Best Ask。注意：若有交易者提交新的Market Order时，则会立刻与LOB中记录的Limit Order发生匹配，也会导致LOB的更新。（2）当有交易者取消了尚未执行Limit Order也会引起LOB的更新，图3中展示的是限价为20.6的Limit Ask Order被它的提交者给取消了，所以LOB就会删除这一指令的记录，导致20.7从Level-2 Ask变成了Level-1 Ask（Best Ask）。

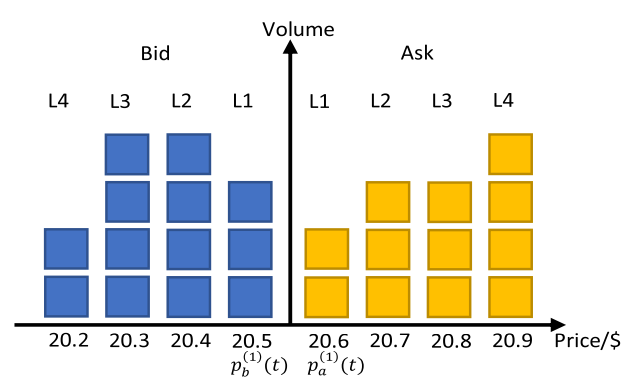


图1 LOB数据示意图

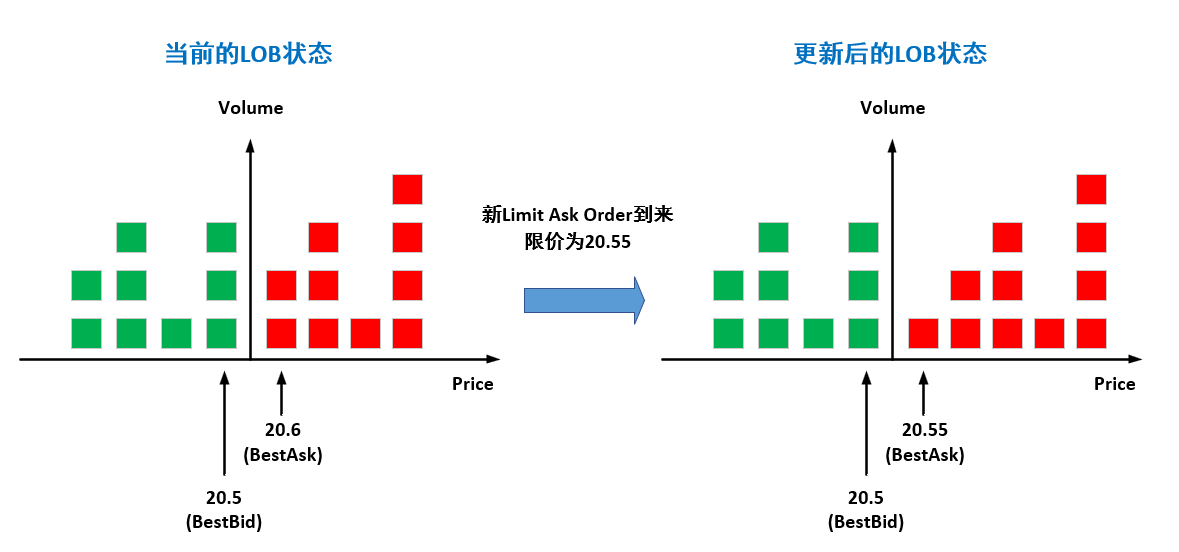


图2 新Limit Order到来导致LOB更新

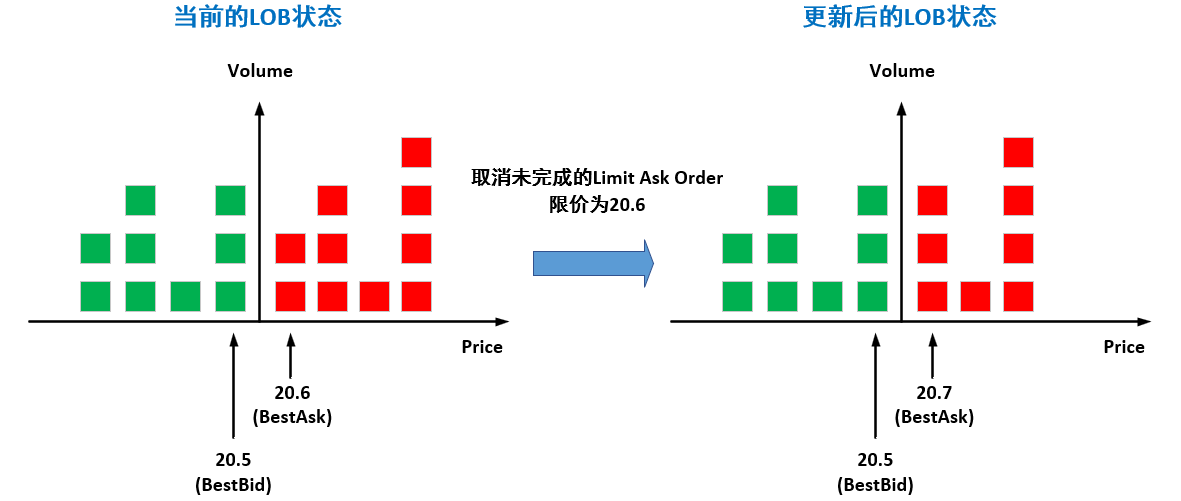


图3 取消未完成的Limit Order导致LOB更新

### 1.1.3 研究目的和意义

在基于LOB匹配机制的金融市场中，**Best Ask**是市场中交易者愿意出售股票的最低价格。也就是说，你可以买到该股票的最低价格。同理，**Best Bid**就是市场中交易者愿意卖出该股票的最高价格。在此引入一个新的指标**Mid Price：**





图4 LOB中Mid-Price的意义

在高频交易市场中，Mid Price是一项重要的指标，它可以表示资产的一般市场价值，所以人们经常使用这一指标来代表资产价值，例如图4中苹果AAPL股票价格其实就是对应LOB中的Mid Price。

因此，对于量化金融研究者而言，如何准确预测短期内LOB中Mid Price的涨跌情况成为了近年来的热点问题。目前的高频交易研究主要包含两大类：价格预测和优化策略，而价格预测也正是优化策略的基础。因此借助LOB进行短期Mid Price变化预测可以有效指导交易者们合理优化自己的交易策略以获得最大收益，具有重要的实际意义。

## 1.2 国内外研究现状

### 1.2.1 概述

已有的方法主要可以分为两大类：统计模型方法以及数据驱动的机器学习方法。其中前者是传统的主流方法，优点是不需要利用大量的数据来训练，复现较为简单；缺点是这类方法经常会对数据的分布进行一些和实际情况不相符合的假设，从而导致应用到实际的高随机性的股票数据时，预测效果往往不好。例如[1]中所提出的VAR模型以及[2]中提出的ARIMA模型。而今年来随着深度学习的兴起，研究者们更倾向于使用数据驱动的深度学习方法来完成高频交易下短期股票涨跌趋势预测任务，本课题也挑选了部分已有的方法进行了复现。

### 1.2.2 基于统计模型的方法介绍

试图利用统计模型或者随机模型来捕获Limit Order Book中的动态特征，并在此基础上准确预测短期内股票价格涨跌趋势是很多统计学派研究者们所希望做到的。

例如，Gourieroux（1999年）[3]、Bouchaud（2002年）[4]和Smith（2003年）[5]通过套用多种统计模型进行试验挖掘出了Limit Order Book中20多种动态特征，并给出广泛统计特征列表。然而，他们的工作仍然不能将特征统一到单个统计模型中加以使用。可喜的是，Cont和de Larrard在2011年的工作[6]中，填补了上述空白，他们提出使用马尔可夫模型来模拟以Limit Order Book为基础的金融市场运作过程，该模型捕捉了Market Order以及Limit Order的主要特征并在此基础上模拟这些特征对价格走势的影响。但是他们的模型仅仅考虑了LOB中Level-1的特征，即大部分剩余特征并不在考虑范围内。值得一提的是，相同的作者Cont在他的另一个工作[7]中确实考虑到了多个Level的Price以及对应的Volume信息，他假定所有即将到来的市场事件（Limit Order，Market Order以及Cancel Order）都遵循独立的泊松过程，以此来预测未来的MidPrice。受到Cont和de Larrard使用马尔可夫模型成功的启发，越来越多的学者开始研究具有随机驱动项的类马尔可夫模型，如向量自回归模型（VAR）[1]和自回归综合移动平均模型（ARIMA）[2]。但是这些模型为了避免过多的参数空间，通常依赖于人工创建的数据特征，但是以此类特征为基础的统计模型大多不具备泛用性，它们往往在某几支股票上表现很好但是在其他股票上表现很差。

总的来说，在2015年之前，使用传统的统计模型来理解Limit Order Book动态过程的研究工作还较多，近年来则是大大减少了。其根本原因就在于这类方法要么大量使用一些人工创建的特征，要么对数据或者市场中事件的分布进行一些和实际情况不相符合的假设，从而导致模型难以应用到实际的高随机性的股票数据。

### 1.2.2 基于深度学习模型的方法介绍

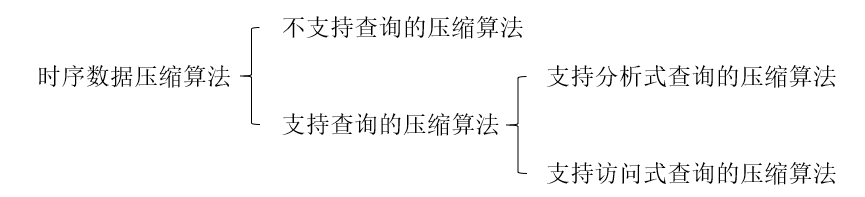


图1-1 时序数据压缩算法分类

如图1-1，已有的时序数据压缩算法可分为支持查询的压缩算法和不支持查询的压缩算法。

不支持压缩数据查询的算法是指压缩数据不支持查询，即必须要全部解压数据才能访问的算法，如经典的应用在Facebook高吞吐实时系统中的Gorilla[5]算法、由Google基于LZ77思想实现的Snappy[6]算法，和阿里巴巴团队开发的两阶段数据压缩框架AMMMO[7]。此类算法或专注于提高压缩率，或专注于提高压缩速度，没有考虑压缩数据被频繁查询的场景，适用于数据存储或传输、访问少的情况。

相对地，支持压缩数据查询的算法可以从压缩数据中提取信息，如计算均值、方差、相关系数等，或对压缩数据进行随机访问。它们注重赋予压缩数据应对查询的能力，适用于数据被频繁访问的场景。按照访问数据的方式可以大致分为两类：支持分析式查询的压缩算法和支持访问式查询的压缩算法。

支持分析式查询的算法从压缩数据中提取统计信息，典型算法为Plato[8]算法。该算法基于有损压缩，将时序数据分为定长或变长的段，并为每段做函数拟合（如用一阶多项式拟合近似直线的数据），由于存储函数的代价远小于存储原数据的代价，达到了压缩的目的，并且存储的函数适用于未来的分析式查询，因此可对不同类型的分析操作给出带error-bound的近似结果。

支持访问式查询的算法适用于频繁的压缩数据访问，如提取某一段数据或某一点的数据。许多经典的时序数据压缩算法需要解压数据才能实现数据访问。[9]提出了一种通用的方法，可以为常用压缩器（如基于LZ编码的压缩器）赋予随机访问的功能，但牺牲了一定的压缩率。[10]实现了一种支持随机访问的时序数据无损压缩策略，基于推广数据去重的思想和字典压缩技术，解决了非常相似的数据块不会被去重算法处理的问题。

## 1.3 该领域存在的挑战及本文的主要研究内容

### 1.3.1 该领域存在的挑战

传统的数据压缩算法往往不考虑对压缩数据进行查询的场景，从而导致对压缩数据的部分访问也需要解压全部的数据。支持查询的数据压缩算法通过牺牲压缩率或数据的准确性支持了对压缩数据的访问或查询，因此，该类算法存在的挑战有两点：

1）对于牺牲数据准确性换取查询能力的算法，通过误差界分析技术尽可能降低误差界，提升数据访问的精度；

2）对于牺牲压缩率换取随机访问能力的算法，通过选取合适的参数尽可能地减少压缩率的损失，并提高数据随机访问的效率。

### 1.3.2 主要研究内容

本次毕设的主要研究内容为在边缘硬件上实现时序数据压缩算法，且能对压缩数据在不解压全部数据的情况下进行查询。研究分为以下几个阶段：

(1) 首先需要广泛阅读文献，了解不同类型的时序数据压缩算法的特点，并结合压缩查询的需求选定合适的算法。

(2) 其次，需要将选定的算法在Linux环境下实现，由于边缘硬件运行在基于Linux内核的操作系统上，因此选定其作为初步实现的环境。

(3) 对算法做出改进也是很重要的研究内容，可根据前期阅读文献的积累，对算法进行优化；另外由于边缘硬件对算法提出的新挑战，需要使算法适应边缘硬件做出改进，如将批处理算法改为在线算法等。

(4) 熟悉边缘硬件的操作系统与数据接口，并移植算法。

(5) 最后，需要利用实验来评估算法在边缘硬件上的运行性能。由于本次毕业设计是实现并优化算法而非设计算法，因此实验部分会更多地进行已实现算法在不同数据集、不同配置下的纵向比较。

## 1.4 章节安排

本论文共包含四章，各个章节内容安排如下：

第1章为绪论，首先介绍了课题的背景以及研究的目的和意义。从整体上阐述此毕设存在的重要性。然后，介绍了国内外在时序数据压缩查询方向的研究现状，先是介绍了普遍用于压缩算法的压缩技术，然后对该研究方向的算法进行分类并一一介绍。最后，阐述了在该领域存在的挑战以及本次毕业设计的主要研究内容。

第2章为Plato算法的实现与优化，主要介绍了一种基于有损压缩、可对压缩数据进行分析式查询的算法的实现与优化过程。该章分为四节，其中二三节为主要内容，介绍了该算法的离线压缩与在线查询实现过程，并使用误差向量的正交投影性质对误差界做改进。

第3章为GD算法的实现与优化，主要介绍了一种基于无损压缩、可对压缩数据进行随机访问的算法的实现与优化过程。该章分为四节，其中二三节为主要内容，介绍了该算法的压缩、解压缩与随机访问实现过程，并对随机访问的代价进行了分析。

第4章为实验的设计与分析。这一章主要是对前两章的两个算法在各个方面的性能指标进行测评。首先介绍了实验平台：华为的轻量级边缘硬件Atlas，然后分两部分对两个算法分别设置实验。首先介绍了各自的数据集的含义及预处理过程，给出数据集的统计信息，然后分别进行实验统计各自的指标，如压缩率、误差界、随机访问代价等。

最后，在结论中，我们对本次毕业设计所做出的贡献进行总结，对所实现并优化的两个时序数据压缩查询算法进行了汇总，并对实验结果进行了综合的评价以及对未来该领域工作的进一步展望。

# 第2章 Plato算法的实现与改进

## 2.1 引言

支持分析式查询的时序数据压缩算法使压缩数据可应对分析式的查询，我们选定该类算法中的Plato算法进行实现并优化。Plato算法将时序数据进行分段压缩，并对每段数据从预设定的函数族中选定最小化欧氏距离的函数进行拟合，通过存储每段时序数据的拟合函数和误差度量，达到了压缩的目的，并且可对未来的查询提供带error-bound的结果。

在本章中，我们主要阐述了实现Plato算法的过程和对其进行的优化。图2-1说明了Plato算法的运行流程，在数据插入时（为了支持error-bound的查询，该阶段为离线处理），给定的时序数据被分段并压缩，该阶段算法会在预设定的压缩函数族中为每段时序数据选定最佳的估计函数。压缩后的数据库为每段时序数据存储估计函数的参数，参数所占的空间远小于原数据。在更复杂的版本中，时序数据分段和估计被混合在一起[10][11]，导致了被分割后的时序数据段是变长的。在数据查询阶段，给定查询q，Plato会对压缩数据进行分析并给出近似结果，该结果可能用到多段时序数据。

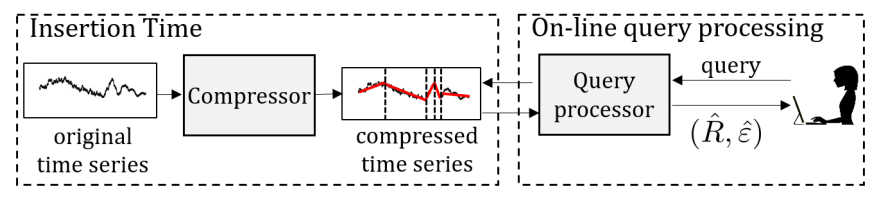


图 2-1 Plato近似查询：离线压缩与在线查询

## 2.2 Plato数据压缩

### 2.2.1 时序数据分段

时序数据分段算法将时序数据分成不相交的段，现存的优秀的时序数据分段算法可按如下分类：

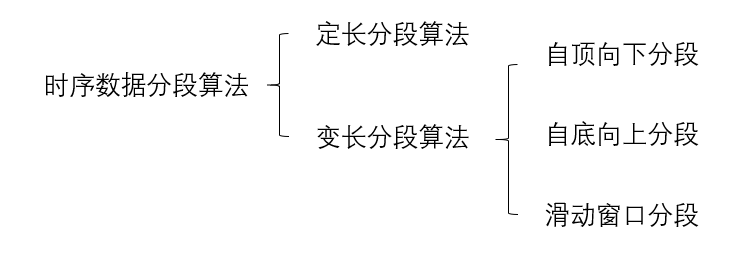
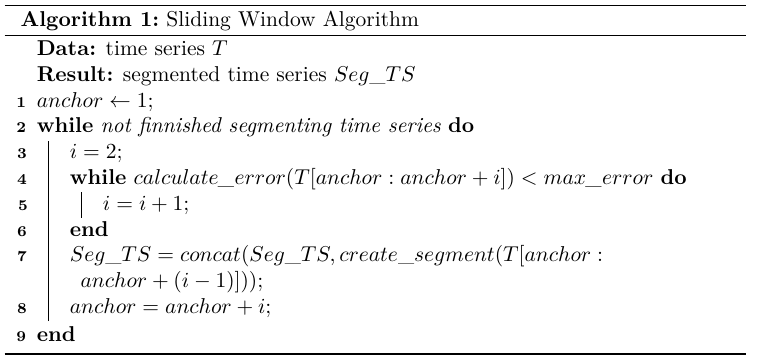


图2-2 时序数据分段算法分类

定长分段算法通过固定大小的时间窗口将时序数据分为固定长度的段；变长分段算法可分为自顶向下分段算法[12][13]、自底向上分段算法[14]和滑动窗口分段算法[10]，由于滑动窗口算法被证明有更高的效率，在本次设计中选择该算法作为时序数据分段算法的实现，算法伪代码如下：



第1行初始化anchor变量作为滑动窗口左边界，第3到第7行不断扩张右边界直到估计误差大于max\_error，并将左右边界间的数据作为一个数据段，如此循环直到左边界到达最后一个时序数据，算法运行结束。

### 2.2.2 估计函数选择与存储

将时序数据分段后需要进行拟合，具体流程为：

**1) 用户选定函数族** 用户选择用于压缩分段后的时序数据的函数族，本次毕设预设了拟合效果好、error-bound更小的一阶多项式函数族。

例2-1 给定时序数据，在选定一阶多项式函数族的前提下，Plato会选择作为拟合函数，因为该拟合函数与数据的欧氏距离最小。

**2) 拟合函数选择** Plato从函数族中选择最合适的函数，选择的指标为最小化原始值与估计值的欧氏距离。即如下的函数：



一旦拟合函数被确定，Plato会存储其函数表示，即函数的参数。

### 2.2.3 误差度量

除了拟合函数，Plato也会为每段时序数据存储额外的误差度量，三种误差度量见表2-1：

表2-1 拟合误差度量分类

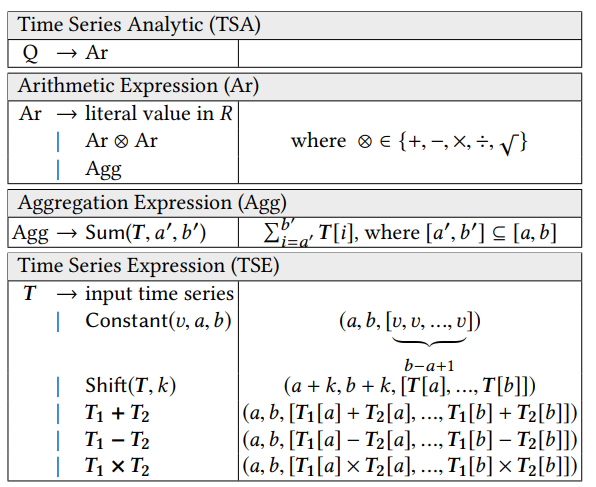
|  |  |
| --- | --- |
| Error measures | Comments |
|  | 估计误差的二范数 |
|  | 估计值的二范数 |
|  | 估计误差的绝对值 |

还是例2-1中的例子，对于时序数据和拟合函数，，，。

## 2.3 Plato数据查询

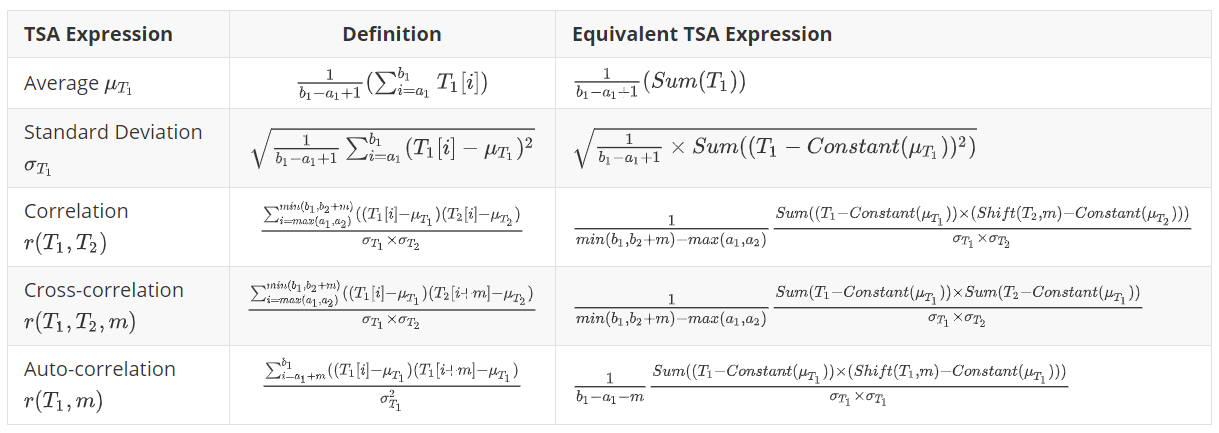
Plato通过为每段时序数据存储的拟合函数及其误差度量为后继查询提供error-bound的结果，给定时序数据和查询q，令R为q的精确结果，为在压缩数据上执行q得到的近似结果，表示查询q的真实误差，Plato的目标是估计出的上界，使结果具有100%置信度，具体支持的操作见表2-2。

表2-2 时序数据分析操作



通过时序数据分析操作，Plato可以表达出单时序数据统计量（如均值、方差）和多时序数据统计量（如相关系数、互相关），支持的统计量见表2-3。

表2-3 时序数据分析操作可表达的统计量



### 2.3.1 误差传播

从表2-2中可知，查询q可通过算术操作关联多个聚集值，如，其中，为了为查询q提供确定的误差界，关键是要计算出每个聚集值的误差界，一旦每个聚集值的被计算出来后，就可以通过组合误差界得到最终结果的误差界，表2-4总结了算术操作误差传播的计算方法。

表2-4 算术操作误差界计算方法

|  |  |
| --- | --- |
| Operator | Error guarantee |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

接下来讨论单个聚集值的误差界的计算方法，给定聚集值和压缩后的时序数据，要计算，根据T是否为输入数据分两种情况讨论：

1. 如果T是输入数据，；

(2) 如果T是通过时序数据运算操作（如）产生的数据，则可被表示成表达式树。图2-3展示了相关系数用聚集量表示的表达式树，我们对该树的叶节点（输入时序数据）加入误差度量，然后根据一定规则将误差从叶节点传播到根节点作为该聚集量的误差界。

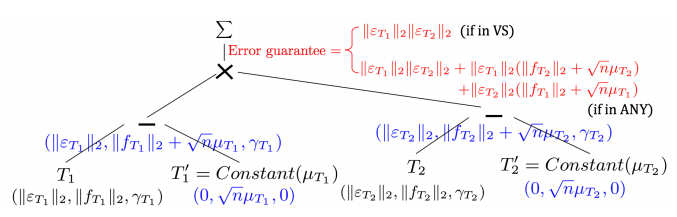


图2-3 用聚集量表示的相关系数表达式树

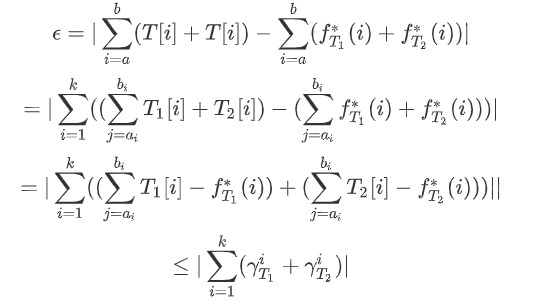
### 2.3.2 时序数据操作的误差界分析

下面对表2-2中的时序数据分析操作进行误差界分析。

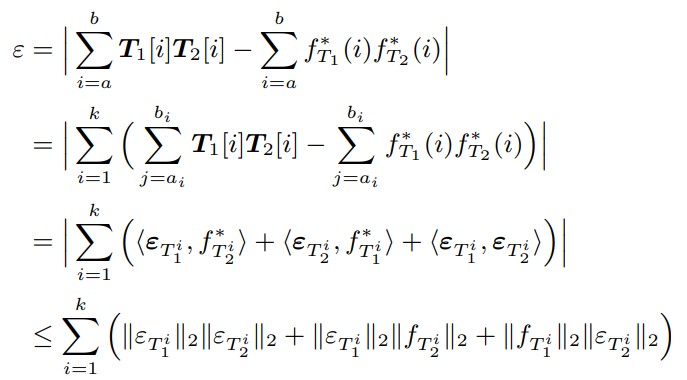
对于操作，拟合函数。Plato存储的误差度量为，，即对常数时序数据的聚集值误差为0。

对于操作，由于只是对时序数据做平移操作，并没有进行其他的修改，因此操作后的时序数据误差度量不变。

对于操作（同理）。给定时序数据，，令，表示和的误差度量，可推导出的误差界如下：

 (2-1)

对于的操作，要分为和段对齐（即与的范围相同）和段不对齐的情况讨论。首先讨论较简单的段对齐情况，此时的误差界如下：

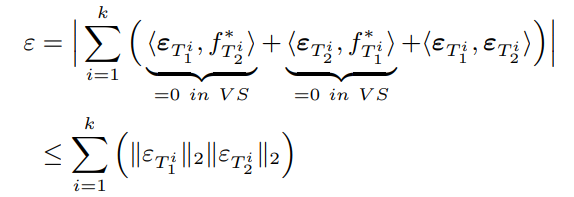
 (2-2)

### 2.3.3 使用正交投影性质改进误差界

Plato算法通过使用正交投影性质使进一步降低，正交投影性质见引理1。

**引理1（正交投影性质）：令F为一个形成向量空间VS的函数族，且是该族中选出的用来拟合时序数据T的函数，则是时序数据T在F上的正交投影。**

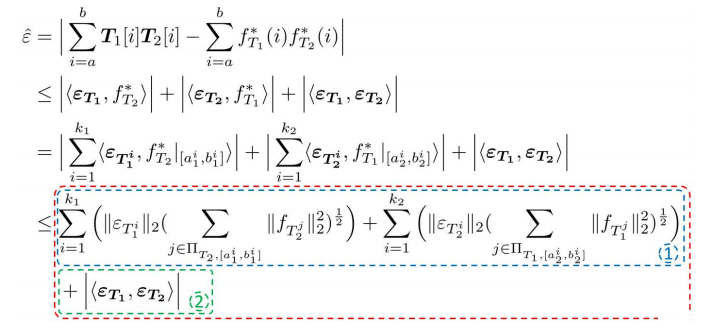
引理1暗示了与任意正交，即，所以给定两个段对齐的时序数据段和，如果拟合函数和都属于向量空间VS的话，就有和。根据此性质，我们可以对进行改进：

 (2-3)

与公式2-2相比，三项误差中的两项被消除，误差界大大减少。

例2.2 给定时序数据，拟合函数族为一阶多项式，最小化欧氏距离的拟合函数，则为在上的正交投影，误差向量 。基于引理1，对于任意拟合函数，都有。

### 2.3.4 段不对齐的情况下改进误差界

两组时序数据经过分段后通常是段不对齐的，原因有以下两点：1. 变长分段算法具有更好的压缩效果，但该算法对不同的时序数据产生的分段通常是不对齐的，所以多时序数据操作如相关系数、互相关等会使用段不对齐的两个时序数据，此时需要对误差界做进一步优化。2. 即使原数据是对齐的，某些计算指标（如互相关、自相关）会对数据执行平移操作，导致平移后的数据不对齐。对于段不对齐的时序数据，我们可以通过类似的分析得到的上界

(2-4)

其中 表示在区间上的投影向量，表示中任意与区间相交的段。此时由于与相交的段可能不止一个，它们构成的函数并不一定属于函数f所在的族F，见图2-4。

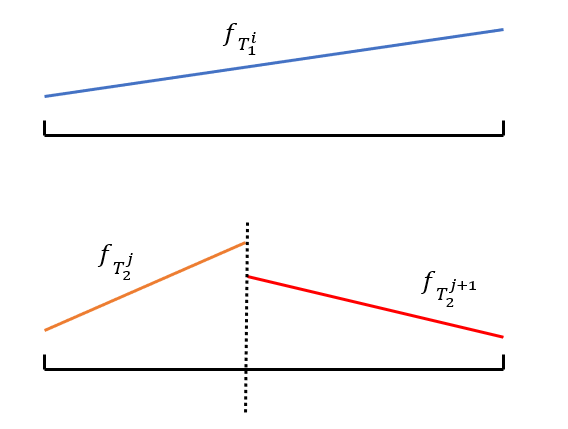
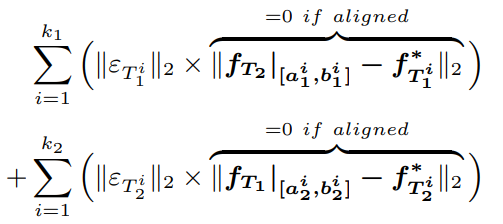


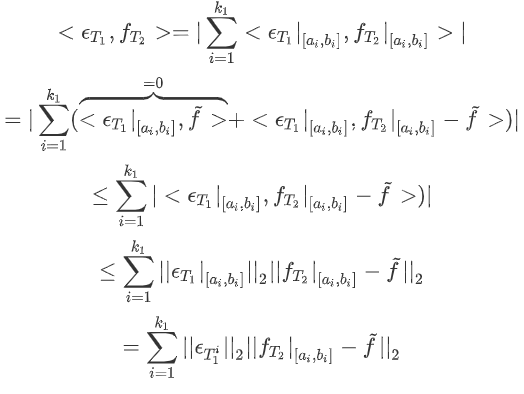
图2-4 与相交的段可能不止一个

此时便不能使用正交投影性质将公式2-4中的①降低为0，但可对其进行尽可能地降低，得到如下表达式：

 (2-4①)

### 2.3.5 对误差界的进一步改进

通过观察公式2-4①，可发现其将原式中的降低为，即通过在二范数内减去对应段的估计值向量降低error-bound，我在其工作的基础上通过改进证明步骤进一步降低了error-bound，证明如下（以为例，同理）：

 (2-5)

其中表示任意，即对用任意同族函数拟合得到的误差的二范数，该值远小于公式2-4①中的值。

## 2.4 本章小结

在本章节中，我们对Plato算法的离线压缩、在线查询过程进行了分析。

Plato算法通过将时序数据分段，为每个段从拟合函数族中选择最合适的拟合函数，并存储拟合函数的参数与误差度量达到了压缩的目的，该过程由于要使用滑动窗口算法进行分段，通常是离线进行的。通过保存每个数据段的误差度量，Plato支持多种时序数据分析操作，包括加、减、乘、聚集等，将时序数据聚集成聚集量后会为每个聚集量计算误差界，最后按聚集量的误差界计算规则得到最终结果的误差界。作者使用正交投影性质对在段对齐和段不对齐情况下分别进行了优化，我在其优化的基础上改进了段不对齐情况下的误差界，使其在原来的基础上大大降低。

# 第3章 GD算法的实现与改进

## 3.1 引言

支持访问式查询的时序数据压缩算法使压缩数据可应对访问式的查询，我们选定该类算法中的GD算法[15]进行实现并优化。传统的数据去重算法会将仅有一点差别的块视作不同块，显然这浪费了块之间潜在的相似性。GD算法利用了推广的去重思想，通过提取块之间相同的部分为base并进行去重，不同的部分为deviation按原数据存储，更好地利用了不同块之间的相似性，并以非常低的代价支持随机访问。

在本章中，我们主要阐述了实现GD算法的过程和对其进行的优化。图3-1说明了GD算法的运行流程，在压缩阶段，算法对时序数据进行分块，并通过转换函数对块进行重排列、提取base和deviation，最终base被存储在字典中，deviation存储在压缩后的数据中；在查询阶段，通过读取字典和参数可对原数据进行随机访问。

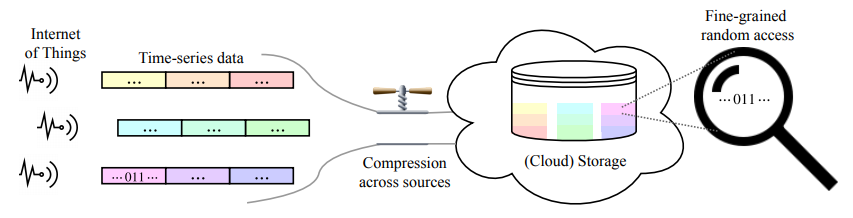


图3-1 GD算法运行流程

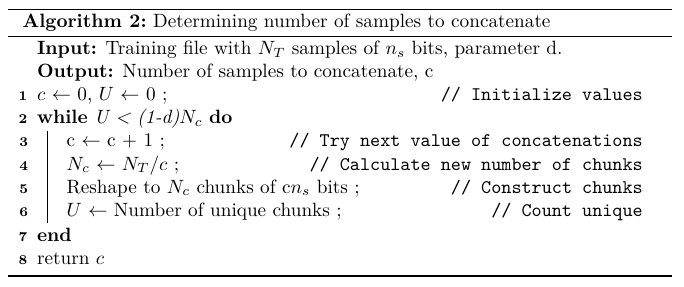
## 3.2 GD数据压缩

GD算法数据压缩分为两个阶段，预处理阶段将待压缩数据的开头一部分数据作为训练数据，从而推断出压缩算法的关键参数。压缩阶段通过参数构建压缩器，对文件以流处理的方式进行一个一个地压缩。

### 3.2.1 预处理

预处理阶段通过利用开头一部分的压缩数据作为训练数据，以确定压缩过程的关键参数，这部分数据可以是文件流中的第一个文件，且必须具有代表性，即能反应数据的整体结构，首先设定每个数据由个比特位表示。

该阶段首先要计算的参数是构成数据块所需要数据的个数c，最佳的c值取决于相邻数据之间的相关性，如果设置的较小会减少编码base ID所需的位数，但可能对块相似性的利用不够充分；如果设置的较大会降低base相同的可能性，因为需要在更多的位数上相同，合适的c值应该从数据中推理出来，该算法的伪代码如下，随着算法的迭代，构成块的数据越来越多，直到几乎每个块都不同（允许比例为d的块相同，实验中该值取0.01），算法终止时c的值即每个块中数据的个数，每个块包含个比特位。

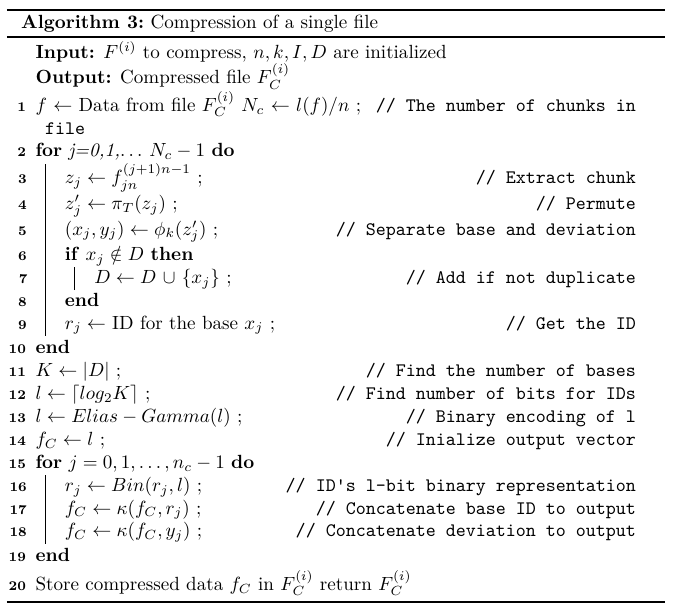


下一个要确定的参数是块中构成deviation的位数，为了使相同的base数量尽可能多，我们想让deviation包含和块中其他位关联最小的位。举个例子，10001、10010、10000、10011这四个块中，显然最后两位与其他位关系最小，因此最适合做deviation。这里通过估计位之间的互信息来表示“关联”的大小，但估计互信息通常是很困难的，因此采用皮尔森相关系数来替代，根据[16]的研究，皮尔森相关系数携带了和互信息相同的信息。将每个bit到其他bits的相关系数平均值按降序排列，越靠后的bit越适合作为deviation，为了计算最适合的deviation位数，不考虑参数的存储先粗略地表示出压缩后文件的大小：

其中K表示不同base的数量，N表示块的数量，首先令k=n，然后每次迭代令k减一，直到即估计大小达到了局部最优值。在实验过程中，我发现算法有时会过早到达局部最优值，导致最终压缩效果很差，因此我将该迭代算法的结束条件优化为，避免了很小的扰动就使算法终止。最终将作为deviation的n-k个bit的索引存入*I*中，并在原数据块中将其移动到块的最后面。

### 3.2.2 压缩

该阶段对数据进行压缩，压缩时文件可以以文件流的形式，即一个一个地到达，如果文件不能被划分为整数个块，最后一块使用零填充。算法的伪代码如下，第一趟扫描通过维护base字典，对文件中的每一个块分别进行重排列、提取base、查询字典、分配base ID；第二趟扫描将base ID和deviation拼接存储，建立压缩后的块。第一趟处理后，会计算编码base ID所需的位数，其中为字典的当前大小，并将其用Elias-gamma编码存储在文件头部。



最终的存储格式见图3-2，原文件分块后进行压缩，文件流经压缩后生成三种文件：压缩器参数、字典和压缩数据。压缩器参数存储单个数据所占比特数，每个数据块包含的数据个数和重排列索引；字典包括所有数据块中不同的base；压缩数据包含指向字典中对应条目的base ID，和块之间不同的部分deviation。

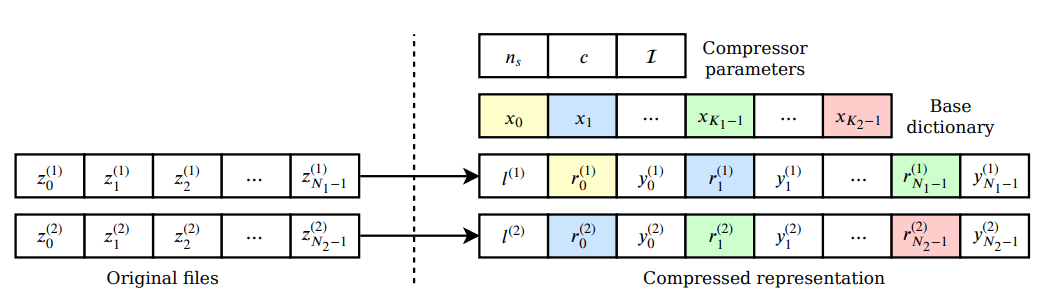


图3-2 原文件格式与压缩后的文件格式

### 3.2.3 压缩文件大小分析

我们使用Elias-gamma编码来编码整数，Elias-gamma编码是一种简单的前缀编码方式，任何整数a都可以使用位比特表示，需要存储的整数包括每个数据所占比特数，每个数据块的数据个数和重排列字典中的索引，需要的总比特数为

压缩后的单个文件大小表示为，该值与原文件中的数据块个数和编码base ID所需要的比特位数有关，所以压缩后的文件大小可表示为

其中，第一项为编码的代价，第二项为编码原文件每个数据块的代价，其中包括长度为的base ID和长度为的deviation。

存储base的字典的代价也要计入，可表示为

其中为字典的大小

最终，压缩后的数据总大小为

对于一般的数据来说，(3-2)的开销都可以忽略不计，真正的挑战在于选择合适的参数平衡(3-3)和(3-4)的开销，这也是预处理阶段的目标。

## 3.3 GD数据解压缩与随机访问

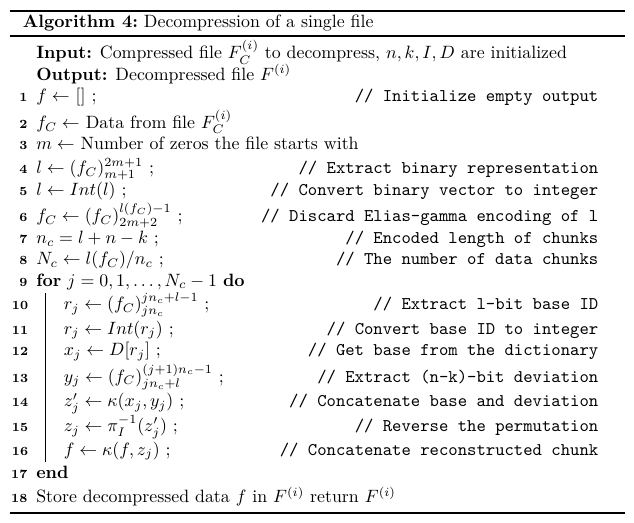
GD算法解压缩可对压缩数据进行完全的解压缩，随机访问是推广的去重压缩方法提供的特性，允许对原数据的任意比特位进行低代价的随机访问。

### 3.3.1 GD数据解压缩

GD数据解压缩分为两个阶段，首先要恢复压缩器状态，然后对文件进行解压缩。

恢复压缩器状态需要先读取预处理阶段计算的参数和，然后可计算出数据块的大小为个比特，base的长度为，最终读取字典。由于字典的组织形式是可预知的，字典中第j个连续k比特块对应的base ID为j-1，所以后续的解压缩可以直接根据base ID找到对应字典中的base。

解压缩阶段的算法伪代码如下，压缩文件开头的Elias-gamma编码首先被恢复，得到base的位数，随后每读取到位作为base ID，紧随其后的位作为deviation，由于已被恢复，可通过中存储的重排列索引将deviation恢复至原位，从而得到原来的数据块。



### 3.3.2 GD随机访问及代价分析

推广的去重压缩方法允许对原数据的低代价随机访问。首先必须读取一部分数据以恢复压缩器状态，即预处理阶段的参数和，并计算出和，恢复这些数据需要访问的比特数为，即公式(3-2)的值，该部分的代价是独立于所存储数据的大小的，即所有对数据的访问都存在的代价。下面将按照数据访问方式分别计算代价。

访问单个数据块时，令我们要访问的块为，即第i个文件的第j块，首先要读取文件头部的，即编码base ID所需的位数，然后计算出该块的起始位置为

结束位置为

通过块中前位，即base ID，可找到字典中对应原base的位置，恢复出原块后再根据中存储的重排列字典进行位置恢复，即得到重排列前的块，总访问比特数为

仅仅比未压缩的数据多访问了位，如果参数已被恢复，那么仅需位。

访问某个比特位的代价取决于该位是base还是deviation，假设我们要访问第i个文件的第a个bit，在该文件中，该bit应定位到块，在该块中，a应位于位置a mod n。

**访问deviation bit** 如果，那么该bit属于deviation，如果是的第j个元素，则该bit是deviation的第j个元素，该位置为，总访问bit数为：

**访问base bit** 如果，那么该bit属于base，首先读取块前的位以确定base ID ，于是可确定要访问的base在base字典中起始位置为，结束位置为，还需知道有多少个deviation bit原来在前，现在被重排列到了块的后面，该值可表示为

其中表示指示函数，当内部表达式为true时取1，false时取0，所以要访问的bit在字典中位于，总访问bit数为：

## 3.4 本章小结

在本章节中，我们对GD算法的数据压缩、解压缩、随机访问过程进行了分析。GD算法使用推广的数据去重方法，将数据块划分为base与deviation，其中对base使用传统的数据去重方法进行去重，对deviation按原数据存储，且考虑到deviation不总在数据块的最后面，通过皮尔森相关系数估计互信息，将deviation移动至数据块的最后面。推广去重解决了相似块无法被去重的问题，且由于去重算法的特性，可通过压缩参数和数据字典恢复压缩器状态，从而对原数据进行低代价的随机访问，我们对随机访问的代价进行了分析，分析得出在压缩器状态已被恢复时，随机访问的代价非常低。我们还对该算法预处理阶段的终止条件进行了改进，避免了很小的扰动就使算法终止。

# 第4章 实验设计与分析

## 4.1 实验环境

本次实验目的是在边缘硬件上实现时序数据压缩与查询算法，因此使用华为的面向广泛边缘应用场景的轻量边缘设备Atlas 500作为算法部署环境，Atlas 500的系统架构图如图4-1所示，处理器为华为自研海思Hi3559A，在该处理器上运行了一个基于Linux内核的定制操作系统Euler，用于管理硬件资源。由于Euler基于Linux内核，因此为了保证算法的可移植性，采用ubuntu 18.04作为开发环境，c++作为算法实现语言。

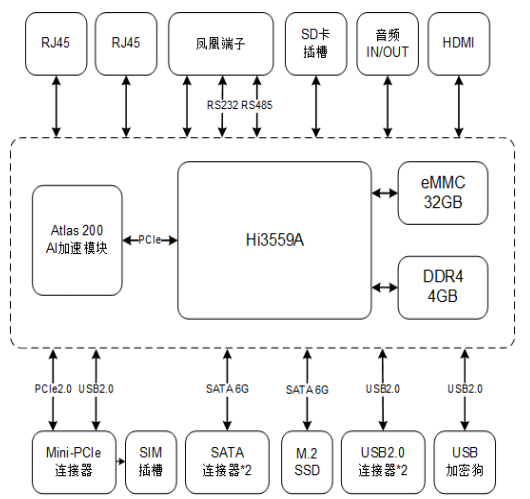


图4-1 Atlas 500系统架构

就算法本身来讲，运行在边缘设备和普通计算机上差别不大，但边缘设备在应用场景上更接近数据的源头，对数据的处理更加及时，因此在边缘设备上的实验设置与在开发环境的实验设置基本相同。同时考虑到边缘设备有限的存储容量和未来要面对的大规模数据的压缩需求，算法要尽可能地具有实时性，本次毕设选定的两个算法均具有离线压缩、在线查询的能力，符合边缘硬件对算法提出的新要求。

## 4.2 Plato算法分析

### 4.2.1 数据集及预处理

在该算法中，我们使用了两种真实数据集：比特币成交记录数据集(HB)和空气质量记录数据集(HA)，表4-1总结了这两种数据集的统计信息。

表4-1 Plato数据集统计信息

|  |  |  |  |
| --- | --- | --- | --- |
|  | 平均数据点个数 | 时序数据数量 | 单位 |
| HB | 4,860,000 | 8 | 分钟 |
| HA | 1,587,258 | 11 | 分钟 |

HB数据集包含了从2012年到2018年每分钟的数字货币成交价格，以逗号为分隔符，csv格式存储，数据中有缺失值。对该数据集进行处理时，我们使用前一个非缺失值去填充当前的缺失值，并将每一个时序数据单独导出，不包含列名。

HA数据集包含了从2011.9.10到2014.9.10圣地亚哥市11个不同的空气质量指标如气压、气温、风向等，数据中无缺失值。对该数据集进行处理时，直接将每一个时序数据单独导出，不包含列名。

### 4.2.2 实验分析

本次实验使用已被证明效果更好的滑动窗口算法[10]作为时序数据分段算法，使用具有正交投影性质，可对error-bound进行进一步优化的一阶多项式作为时序数据段拟合函数。

我们首先通过在HA数据集上执行查询，评估了真实误差（估计值与真实值的误差）和误差界的大小及比例关系，结果见表4-2。如表所示，进行分析的操作包括单时序数据操作：均值、标准差，Plato对该类型的查询给出了接近100%准确的结果；对于多时序数据操作相关系数，Plato给出了相差一个量级的error-bound，对于自相关，Plato给出了同量级的error-bound。

表4-2 时序数据分析操作的真实误差与误差界

|  |  |  |  |
| --- | --- | --- | --- |
| 时序数据分析操作 | 真实误差(True Error) | 误差界(Plato) |  |
| 均值 | 0.00 | 0.00 | 0.00 |
| 标准差 | 0.0221474700 | 0.0221474700 | 0.00 |
| 相关系数 | 0.0041128962 | 0.0626709724 | 14.2376740548 |
| 自相关 | 0.0157846300 | 0.0620285434 | 2.9296799036 |

Plato算法可通过控制分段阶段的误差参数，从而控制压缩文件的压缩率。由于后续实验部分会通过调整参数将压缩文件大小调整到很低，此处压缩率定义采用原文件大小除以压缩文件大小，即压缩率为100表示压缩文件大小为原文件的1/100。为了验证不同压缩率对真实误差与误差界的影响，我们固定查询为相关系数，通过调整参数控制压缩率得到了HB数据集上查询的真实误差与误差界，结果见图4-1。总体来看真实误差在误差界的范围内，但随着压缩率的提高，误差界对真实误差的限制逐渐放松，即二者的差距逐渐增加。误差界在压缩率为10和20时比真实误差高一个量级，在压缩率为50时比真实误差高两个量级，但二者的值作为相关系数的误差都非常小（在水平），因此相比于真实结果0.9999983084几乎可以忽略不计。

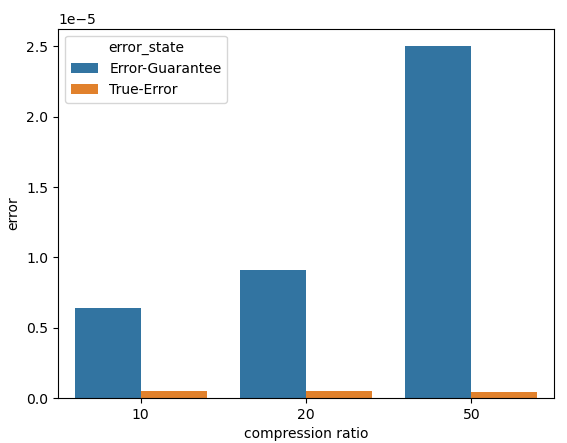
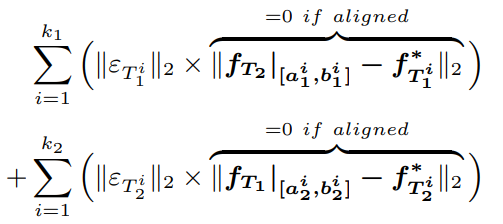


图4-1 HB数据集上的true error与error bound

作者在对段不对齐时的进行估计时使用正交投影性质将公式2-4①优化为了



该式只在与均值相近时才会达到优化的效果，因为此时与同号且同量级，相减后才会在原值基础上大幅降低。但在与相差几个量级，或者与是异号的时序数据时，与相减会导致原值基本不降低，甚至可能不降反增。下面的实验说明了这点，图4-2展示了在HB数据集上两个均值分别为3077.54134和3077.508174的时序数据的相关系数查询结果误差界，此时时序数据均值几乎相等，优化后的误差界没有显著的降低。图4-2展示了在HA数据集上均值为916.8301267和61.85144078的两个时序数据的相关系数查询结果误差界，此时时序数据误差相差一个数量级，优化后的误差界显著降低。

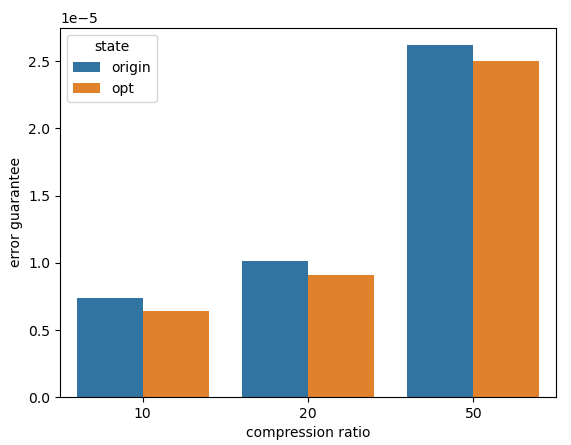


图4-2 HB数据集上的优化前后误差界

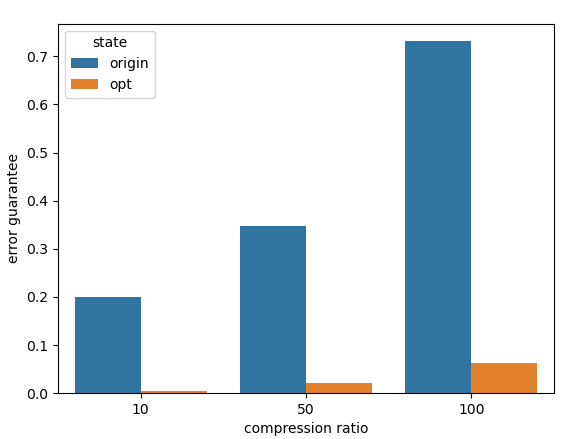


图4-2 HA数据集上的优化前后误差界

## 4.3 GD算法分析

### 4.3.1 数据集及预处理

在该算法中，我们使用了三种数据集，表4-2总结了这三种数据集的信息。

表4-2 GD算法数据集介绍

|  |  |  |
| --- | --- | --- |
| 数据集名称 | 描述 | 大小 |
| ECG | 麻省理工学院心律失常信息数据库 | 93.6MB |
| T-DK | 丹麦8个地点的日最高气温 | 2.1MB |
| Elec | 10个家庭的用电量情况 | 4.0MB |

ECG数据集记录了麻省理工学院收集的心律失常的心电图数据，数据集中包含三种数据：.atr：注释数据，记录了相应心电图信息的种类；.dat：数据文件，按二进制存储；.hea：头文件，按ASCII码存储，包含一些基本的常量。本次实验对.dat文件进行压缩。

T-DK数据集记录了丹麦8个地点的日最高气温，以文本形式存储整数，包含21行的文件头。对该数据集进行预处理时，首先跳过前21行的文件头，然后将剩余数据转换为二进制。

Elec数据集记录了10个家庭的用电量情况，以文本形式存储浮点数。对该数据集进行预处理时，读取所有的浮点数并转化为二进制形式存储。

### 4.3.2 实验分析

为了比较基于推广去重方法的GD算法与传统的压缩算法的压缩率，我们选定两种传统算法与GD算法进行压缩性能上的比较：GZIP算法是DEFLATE算法的实现，应用了LZ77编码和哈夫曼编码的思想；7-zip算法使用了LZMA算法进行压缩，LZMA算法是LZ77算法的扩展。由于无损压缩保证了信息的无损失，其压缩效果通常比有损压缩差，所以后续实验部分的压缩率定义为压缩文件的大小除以原文件的大小，即压缩率为0.1表示压缩文件大小是原文件的1/10。

三种压缩算法对三种数据集的压缩率见图4-3，GD算法的运行参数在柱状图中标明。对于前两个数据集，GD算法的压缩率略低于GZIP算法，高于7z算法；对于最后一个数据集，GD算法的压缩率显著高于GZIP算法和7z算法，可以看到此时deviation的比特数退化为0，即推广的去重算法退化为传统的去重算法，导致了压缩效果一般。

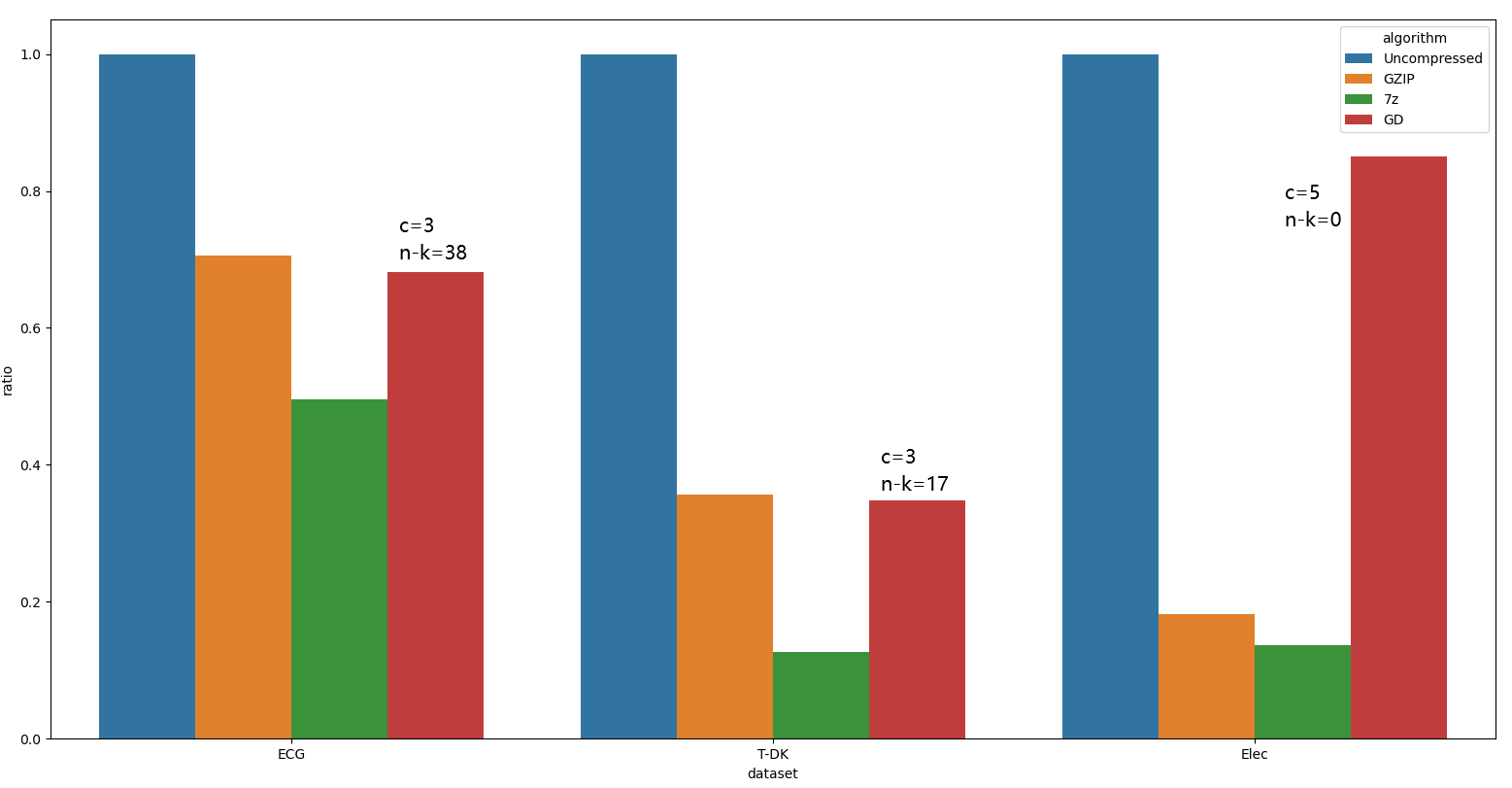


图4-3 三种算法对三种数据集的压缩效果

为了验证预处理阶段的参数选择对最终压缩率的影响，我们通过手动改变参数c和n-k的值，计算了GD算法在T-DK数据集上的压缩率，结果见图4-4。启发式算法选择的参数c=3，(n-k)/n=0.2，可见此时的参数非常接近最小值，说明启发式算法的结果是具有可信度的。

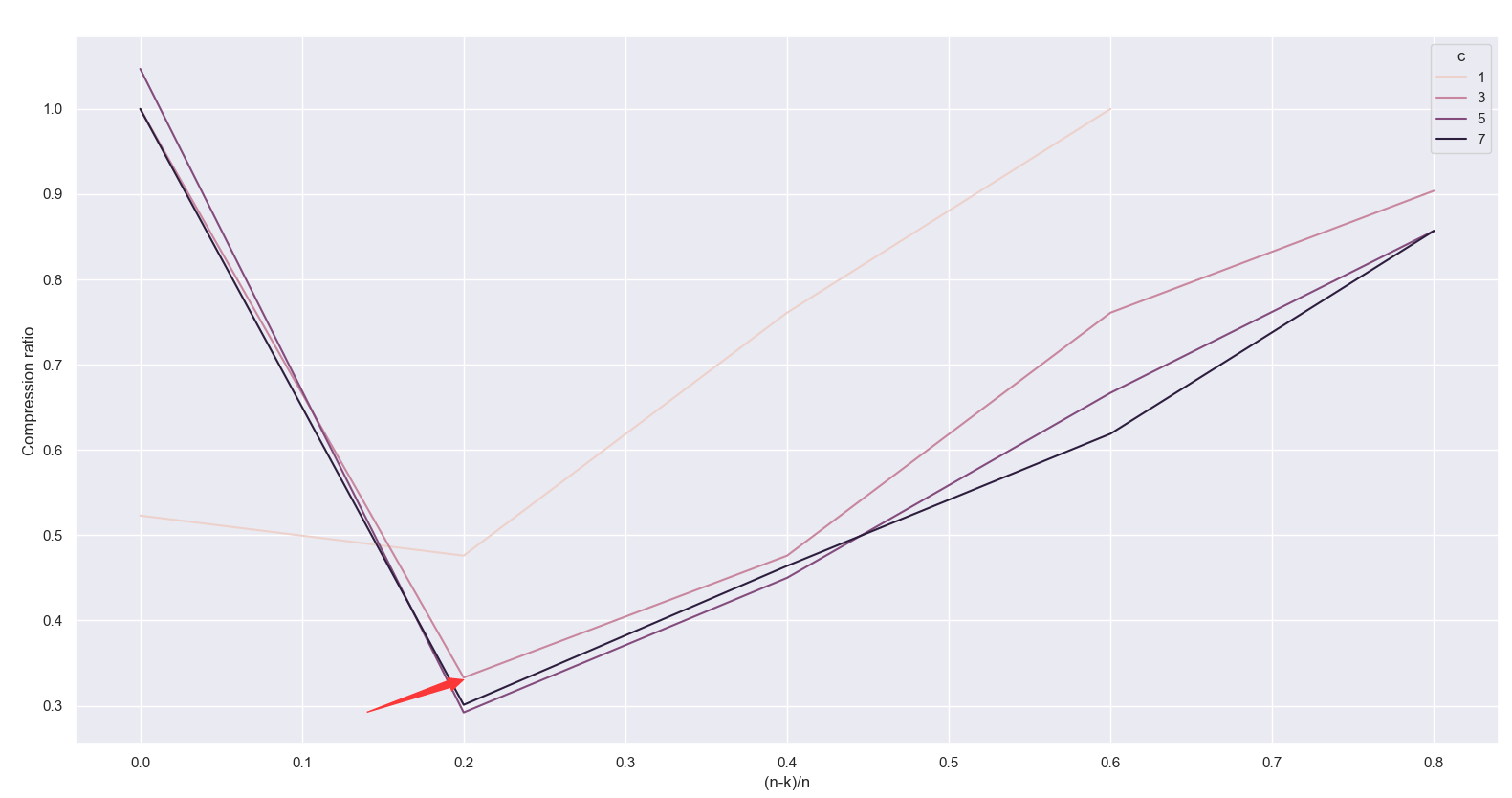


图4-4 启发式算法在T-DK数据集上不同参数的压缩率对比

GD算法相对于其他压缩方法的显著优势在于它提供了对压缩数据的低代价访问能力，相比于其他算法需要全部解压数据后才能进行访问，GD算法通过恢复参数、查询字典对原数据中任意位、任意块进行访问。我们测试了在不同压缩率下访问任意位的代价，该代价以访问的比特位数衡量，结果见图4-5。最坏的情况下（压缩器状态没有恢复），访问单个位需要以访问压缩数据中几百个比特位为代价，但这些代价大多来自于压缩器的状态（参数、字典）；如果压缩器状态已恢复，则访问deviation只需要个位数的访问代价，访问base只需要两位数的访问代价。在牺牲少量的压缩性能的前提下，GD算法提供了远超于传统压缩算法的随机访问性能，对于有着强随机访问需求的场景非常适用。

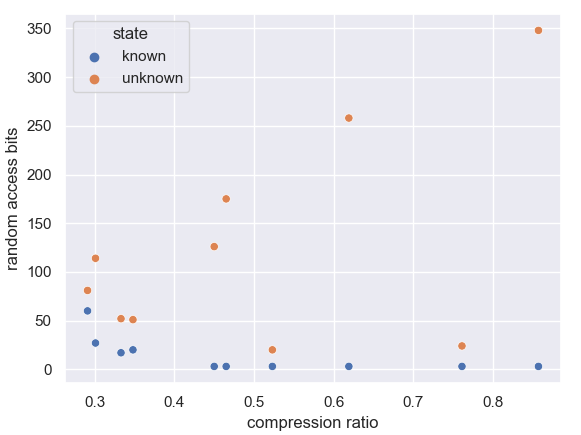


图4-5 不同压缩率下访问单个bit的代价

## 4.4 本章小结

在这一章节中，主要对我们所实现并优化的两个算法Plato和GD进行实验，来测试算法各指标。实验主要安排在二三小节：

第二小节，主要对Plato算法的性能进行测评。首先通过在HA数据集上执行均值、标准差、相关系数等查询，测试了Plato算法给出的误差界与真实误差界的大小，结果表明均值、标准差查询几乎不存在误差，相关系数、自相关的误差较大，但与真实误差在同一量级；然后通过在HB数据集上固定查询为相关系数，测试了不同压缩率下误差界与真实误差界的大小，结果表明误差界虽然比真实误差高1-2个量级，但二者相对于结果都非常小（在水平）。最后比较了使用正交投影性质对误差界做进一步改进前后的误差界变化，结果表明在两段时序数据均值相近时提升较小，但对于两段时序数据均值相差较大的一般情况，误差界有很大改进。

第三小节，主要对GD算法的性能进行测评。首先通过对比GD算法与传统的数据压缩算法GZIP和7z的压缩性能，得出GD算法压缩率略低于GZIP，略高于7z的结论；然后通过手动调整参数并计算压缩率，比较了不同参数下压缩率的大小，结果表明GD算法预处理阶段的参数选择较为合理，比较接近全局最优；最后对GD算法的随机访问效率进行了测试，结果表明压缩器状态未恢复时随机访问需要几百个比特位的代价；但压缩器状态一旦恢复，访问deviation只需要个位数比特的访问代价、访问base只需要两位数比特的访问代价。

# 结 论

在本次毕业设计论文中，我们在边缘硬件上实现了基于有损压缩思想，可对压缩时序数据进行error-bound统计查询的Plato算法和基于无损压缩思想，可对压缩时序数据进行低代价随机访问的GD算法。其中：

1. 对于Plato算法，我们利用了作者提出的误差向量正交投影性质对时序数据乘法并聚集的误差界做了进一步改进。
2. 对于GD算法，通过改进预处理时的算法终止条件避免了预估空间落入局部最优值。

最后通过实验证明了Plato算法给出的误差界具有合理性，且对误差界的改进使均值相差较大的时序数据相关系数查询操作的误差界有很大程度的降低；GD算法的压缩效果比现存的优秀数据压缩算法略差，但牺牲少量压缩率换取的随机访问效果非常可观，且在压缩器状态恢复后的随机访问代价进一步降低。

我们还提出了一些对时序数据压缩查询未来的展望：

1. 随着边缘硬件在智慧城市中的大规模部署，对数据的压缩和查询需求不再拘泥于单机处理，可对分布式情况的压缩与查询做进一步探索。
2. 对于有损压缩，支持近似查询的算法，可以尝试探索拟合函数的更好的性质，对误差界做进一步改进。
3. 对于无损压缩，支持随机访问的算法，可以尝试改进原有启发式算法，使其得出的参数更加合理。

# 参考文献

[1] E. Zivot and J. Wang, “Vector autoregressive models for multivariate time series,” Modeling Financial Time Series S-PLUS, pp. 385–429, 2006.

[2] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, “Stock price prediction using the ARIMA model,” in Proc. 16th IEEE Int. Conf. Comput. Model. Simulation 2014, pp. 106–112.

[3] Gourieroux, C., Jasiak, J. and Fol, G. L. (1999). Intra-day market activity. *Journal of Financial Markets* **2**, 193–226.

[4] Bouchaud, J. P., Mezard, M. and Potters, M. (2002). Statistical properties of stock order books: Empirical results and models. *Quantitative Finance* **2**, 251–256.

[5] Smith, E., Farmer, J. D., Gillemot, L. and Krishnamurthy, S. (2003). Statistical theory of the continuous double auction. *Quantitative Finance* **3**, 481–514.

[6] Cont, R. and de Larrard, A. (2011). Price dynamics in a Markovian limit order market. *SSRN eLibrary*.

[7] Cont, R., Stoikov, S. and Talreja, R. (2010). A stochastic model for order book dynamics. *Operations Research* **58**, 549–563.

[8] Chunbin Lin,Etienne Boursier,Yannis Papakonstantinou.Plato: Approximate Analytics over Compressed Time Series with Tight Deterministic Error Guarantees[J].VLDB,2020,1:1-14.

[9] K. Tatwawadi, S. S. Bidokhti, and T. Weissman, “On Universal Compression with Constant Random Access,” in IEEE ISIT, jun 2018, pp. 891–895.

[10]E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In ICDM, pages 289–296, 2001.

[11] A. Koski, M. Juhola, and M. Meriste. Syntactic recognition of ecg signals by attributed finite automata. Pattern Recognition, 28(12):1927–1940, 1995.

[12] C. Li, P. S. Yu, and V. Castelli. MALM: A framework for mining sequence database at multiple abstraction levels. In CIKM, pages 267–272, 1998.

[13] S. Park, D. Lee, and W. W. Chu. Fast retrieval of similar subsequences in long sequence databases. In KDEX, pages 60–67, 1999.

[14] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In KDD, pages 239–243, 1998.

[15]Rasmus Vestergaard, Daniel E. Lucani, Qi Zhang.A Randomly Accessible Lossless Compression Scheme for Time-Series Data[J].IEEE INFOCOM,2020,1:2145-2154.

[16] Alexander Kraskov,Harald St¨ogbauer,Peter Grassberger.Estimating Mutual Information[J].APS,2008,1:1-16.

# 哈尔滨工业大学本科毕业设计（论文）原创性声明

本人郑重声明：在哈尔滨工业大学攻读学士学位期间，所提交的毕业设计（论文）《时间序列大数据压缩与查询》，是本人在导师指导下独立进行研究工作所取得的成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等作假行为。

本人愿为此声明承担法律责任。

作者签名： 日期： 2020 年 6 月 4 日



# 致 谢

冬来春往，寒暑交替。赴哈尔滨求学的第四年，随着毕业论文的行文至此，大学生活也将画上句号。

感谢父母对我从小的辛勤培养，为了让我接受良好的教育不断奔波，我能进入哈工大并顺利完成学业离不开你们的支持。

感谢我的老师高宏教授，在我刚刚笨拙地进入科研的大门时，是您耐心地做我的领路人，从选题到定稿的每一个阶段都少不了您的悉心指导。

感谢原声吉他社的小伙伴们，在我为论文进展缓慢而烦闷时，是你们陪我放松心情，整理好心态重新面对挑战。

最后，感谢所有成长路上给予过我帮助与温暖的亲人、朋友们，我将不负你们的期望，心怀感恩继续前行！