学校代码： 10246
学　　号： 15210240099

硕 士 学 位 论 文

（专业学位）

基于约束优化的缺失能耗数据填补方法与策略

Method and Strategy of Filling Missing Energy Data Based on Constraint optimization

院　　系：　计算机科学技术学院

专业学位类别(领域) ：　计算机技术

姓　　名：　曾 彬

指 导 教 师：　丁向华　副教授

完 成 日 期：　2018 年 3 月 5 日

指导小组成员名单

顾　宁　　教　授

张　亮　　教　授

卢　暾　　副教授

丁向华　　副教授

目录

摘要

随着节能意识的不断增强，节能减排成为了各地区的重点工作。国家要求建立大型公共建筑节能监管体系，为节能改造提供数据支持，实现节能规划。其中，校园节能监管体系对于节约型校园的建设尤为重要。由于网络中断和设备损坏等因素，能耗数据的缺失在所难免。然而，现存的填补方法存在缺陷，无法满足填补区间总和为固定值的约束条件，因此需要对现有方法进行改进。本文的主要研究工作如下：

一. 提出了基于约束优化的缺失能耗数据填补方法。利用 KNN 的基本思想，以 DTW 作为相似性度量，然后将问题转化为连续等式约束优化问题，最终利用拉格朗日乘数法，将缺失区间内总能耗，分配到各个缺失点，完成填补。

二. 基于复旦大学节能监管平台的能耗数据，设计相关实验验证本文的缺失能耗数据填补算法的精确性和有效性。以 MSE、MAE 和 MRE 作为评估指标，与现存填补方法进行对比，实验结果表明，本文提出的缺失数据填补算法具有更高的填补精度和更好的效果。

三. 提出了面向复旦大学节能监管平台的缺失能耗数据填补策略。详细分析复旦大学节能监管平台的数据特点，并针对不同的缺失情况，制定相应的填补策略。

四. 设计并实现缺失能耗数据填补工具。根据复旦大学节能监管平台的需求，使用.Net 开发平台，开发 Winform 窗体程序，利用多线程技术，实现对复旦大学节能监管平台的定时缺失数据检测和定时缺失数据填补。

关键字：缺失数据填补、连续等式约束优化、拉格朗日乘数法、填补策略

中图分类号：TP3

Abstract

With the increase of energy-saving awareness, energy conservation has been the focus job in lots of areas. For providing data to energy-saving project and accomplishing the energy-saving plan, our country requires establishing a large-scale public buildings energy-saving monitor system, and campus energy-saving monitor system is one of the most important parts of it. Because of the unstable network and the broken devices, the missing of energy data is inevitable. However, the existed methods of filling missing data are flawed. Those methods do not consider the constraint that the sum of filling result should be a certain value. So it is necessary to improve those methods to match the new need. The research work of this paper mainly includes the following four parts:

First of all, this paper proposes method of filling missing energy data based on constraint optimization. It uses the basic idea of KNN, with DTW as a measure of similarity. And then, transforms the problem into continuous equality constraint optimization problem. Finally, it uses Lagrange Multiplier Method to distribute the total energy consumption to every missing point.

Secondly, it designs related experiments based on the energy data of Fudan University Energy-saving Monitor Platform, to verify the accuracy and effectiveness of the fill method this paper proposed. It uses MSE, MAE and MRE as evaluation indicators, comparing with existed fill methods. The experiment shows that, the method this paper proposed has better accuracy and effect.

Thirdly, this paper proposes strategy of filling missing energy data oriented Fudan University Energy-saving Monitor Platform. It analyses the data characteristics of the platform in detail, and develops corresponding strategy for different missing situations.

Finally, it designs and implements the tool for filling missing energy data. According to the demand of Fudan University Energy-saving Monitor Platform, it developing Windows Form program in .Net development platform. Use multi-threaded technology to implement functions of detecting and filling missing data in the platform regularly.

Key words: fill missing data, continuous equality constraint optimization, Lagrange

Multiplier Method, fill strategy

The classification method code: TP3

第一章 引言

1.1 研究背景与意义

在能源日益短缺的形势下，节能减排是全球备受关注的主题。我国从“十一五”提出节能减排目标后，节能减排成为了各地区、各行业的重点工作。随着建筑用能占能源消耗比例的不断增高，建筑节能已经成为节约能源中的重要领域[1]。随着社会经济的不断发展，我国的公共建筑总量在迅速增长，单位能耗也在不断增高。因此，国家要求建立大型公共建筑节能监管体系，为节能改造提供数据支持，实现节能规划。其中，校园节能监管体系对于节约型校园的建设尤为重要，是树立节能工作长效机制、激励大众参与、量化节能效果、挖掘节能深度以及拓宽节能广度的基础，是推动节能工作的重要抓手。校园建筑能耗总量大、种类多、影响力大、节能潜力大，是校园节能减排中非常重要的一环[2]。因此，建立校园建筑节能监管体系刻不容缓。在国家住房与城乡建设部、教育部的领导和支持下，全国节约型校园建设工作已逐渐展开。作为全国节约型公共机构示范单位，复旦大学建立了覆盖四个校区的校园建筑节能监管平台。

目前，校园节能监管平台[2]架构基本上可以分为三个模块：数据采集模块、数据存储模块和系统功能模块。由于各平台在数据存储模块存储数据的方式的不同，关于某个时间段内的能耗值的计算主要分为两种方法：

1）单位能耗值累加。将采集上来的电表数据按照最小时间粒度预先处理好，在数据库中存储每个测量点单位时间内的能耗值，当需要统计某段时间内的能耗时，将这段时间内的单位时间能耗值累加。这种计算方法的优点在于，避免对采集的原始数据的依赖，减少连续时间点能耗值之间的关联性；缺点在于，单位时间能耗值的缺失，会导致所有包含该时间点的时间区间能耗统计的准确性。

2）电表读数相减。数据库中存储每个电表各个时间点的读数，这些值随着时间的递增而递增。当需要统计某段时间内的能耗时，将结束时间点的电表读数减去起始时间点电表读数，从而得到该时间段的能耗。这种计算方法的优点在于，当某个时间点的能耗缺失，不对包含该时间点的能耗统计产生影响；缺点在于，当某个时间点的能耗值出现突增或突降，会影响到以该时间点为起点或终点的能耗统计。

其中，数据采集模块是整个平台建设的基础，然而在运行过程中，经常出现

数据长时间缺失的问题，主要有以下几个原因：

1）网络中断造成数据中断。由于网络改动或者网络故障，导致数据传输的不稳定，最终造成数据中断。

2）设备损坏以及维修期间造成能耗数据缺失。智能电表设备都有一定的寿命，当设备损坏时，从发现到维修或者更换期间，无法采集到相关能耗数据，使得该期间的能耗数据丢失。

3）异常数据被剔除。在采集上来的数据中，包含了大量异常数据，例如电力线路拓扑结构的变更而导致的异常数据，对这些异常数据进行清理之后，就会导致产生大量的数据缺失。

建筑缺失能耗数据的填补问题，已经是我国大型公共建筑能耗监管平台普遍存在的问题[3]。如果能耗值的计算采用上述中第二种方法，那么由于一段时间内的能耗值，是电表两个时间点读数的差值，数据的缺失会导致查询能耗值的结果为突增，或者为负值，造成查询异常。随着实时能耗数据的不断采集，如果缺失能耗数据不及时处理，日积月累，将来在更大的数据量的基础上再去填补，将会遇到效率上的问题，因此缺失能耗数据要早发现早处理。并且，填补缺失能耗数据，尽可能地恢复能耗趋势，能够减少数据盲区，提高数据质量，更有利于决策者做出决策，减少误判。与此同时，缺失数据的填补，维护了能耗数据的完整性，减少对后续能耗分析带来不利影响。

复旦大学通过节能监管平台的建设，加强了对能耗的实时监控和管理，并对能耗数据进行分析和异常情况处理，为学校今后对能源进行定额、定量管理提供了准确的、科学的数据。同时，由于上述原因，该平台同样存在大量缺失能耗数据，并且缺失形式多样。对缺失能耗数据的填补，是复旦大学节能监管平台运行维护系统中重要的模块之一。该平台采用电表读数相减法来计算区间能耗值，因此缺失能耗区间的能耗总和是已知的。

基于以上问题和缺失能耗数据的特点，本文提出基于连续等式约束优化的缺失数据填补算法，该算法基于 KNN 的思想，以 DTW（Dynamic Time Wrap，动态时间规整）作为相似性度量，在已知缺失区间和的条件下，对缺失区间进行填补，并与常用的几种填补算法进行对比，包括均值填补、回归填补和三次样条插值填补。最后，根据复旦大学节能监管平台上的数据特点和具体的能耗缺失情况，制定了相应的缺失能耗数据填补策略，节省存储空间，提高算法的运行效率，从而取得良好的填补效果。

1.2 国内外研究现状

由于采集仪器故障和网络传输不稳定等原因，数据的缺失在所难免，不仅仅是能耗数据，包括其他由传感器采集的数据，例如温度、湿度、PM2.5 和降雨量等气象数据，这些数据的缺失也非常普遍。当这些数据发生缺失时，最简单的方法就是利用与其时间相近的数据来估计缺失值，但是这个方法值仅适用于那些比较平滑，且缺失区间小的情况[4]。S.Bennis 等人改进单变量和多变量技术估计水文数据的缺失值[5]。W.Y.Tang 等人对比了多种缺失数据处理方法，对降雨量缺失数据的处理效果[6]。

目前，针对上述类型缺失数据的填补方法，主要有均值插补、线性插补[7]、基于 K 近邻插补[8]和数值填补等，数值填补主要有拉格朗日插值[9][10]和三次样条插值[11]。Hui Chen 等人对单变量回归插补、多项式模型插补、拉格朗日插补和线性插补 4 种方法进行对比，利用这四种方法填补缺失区间大小为 1-6 小时的能耗和天气数据，发现线性插补和多项式模型插补比其他两种方法更加适合填补用能和天气数据，其中线性插补是最简单和方便的方法[12]。Junjun Hu 等人基于缺失区间相邻的已知数据，与缺失数据之间存在丰富的相关信息的假设，提出了基于角度和基于相关性的插补，并与线性插补和拉格朗日插补进行对比，对干球温度数据进行填补，发现线性插补适合填补 1-2 小时的缺失区间数据，拉格朗日插补适合填补 3-8 小时的缺失区间数据，基于相关性的插补适合填补 9-24 小时的缺失区间数据[13]。Liqiang Pan 等人提出 AKE（Applying K-nearest neighbor Estimation），一种基于 KNN 并结合回归模型的缺失数据填补算法，并与线性插补进行对比，对温度和湿度缺失数据进行填补，最终，AKE 算法取得了更好的填补效果[4]。2016 年，Xiuwen Yi 等人对各种填补方法分为两类：考虑单视图的方法和考虑多视图的方法。考虑单视图的方法指 IDW（Inverse Distance Weighting，反距离权重）、线性回归等只考虑空间相关性的方法，以及 SES（Simple Exponential Smoothing，简单指数平滑）等只考虑时间相关性的方法[14]。考虑多视图的方法指那些同时考虑时间和空间相关性的方法，例如 AKE 算法。同时，Xiuwen Yi 等人提出了 ST-MVL（spatio-temporal multiview-based learning）填补方法，这个方法结合经验统计模型 IDW、SES，和数据驱动算法，包括基于用户与基于项的协同过滤算法。在实验中，ST-MVL 取得了良好的填补效果。

随着深度学习[15]的兴起，有研究者试图利用神经网络算法[16][17]对缺失数据进行填补。P. Coulibaly 等人对比 MLP（Multi-layer Perceptron，多层神经网络），TLFN（the time-lagged feedforward network，时滞前馈神经网络），RBF（the

generalized radial basis function，广义径向基神经网络），RNN（the recurrent neural network，复发神经网络）等 6 种不同的神经网络算法，发现 MLP 精度最高，相对其他 5 种神经网络算法效率也更高[18]。叶学勇等人利用 BP 神经网络算法对电力系统的不良数据进行了有效的修正[19]。尽管神经网络算法能够使填补获得较高精度，但是与其他算法相比，神经网络的效率并无法满足大部分平台的要求。Ceylan Yozgatligil 等人将多层神经网络、EM-MCMC（Monte Carlo Markov Chain based on expectation–maximization，基于期望最大化的蒙特卡洛马尔可夫链）、均值插补和 NRWC（normal ratio weighted with correlations，基于相关性加强的正常比例）等方法进行对比，发现多层神经网络精度较高，但是运行速度却是最慢的[20]。

由上述可知，缺失数据的填补方法多种多样，具体选择哪种方法，则需要结合平台数据特点和已有条件，对这些方法进行取舍或者改进。复旦大学节能监管平台采用电表读数相减法来计算区间能耗值，因此缺失能耗区间的能耗总和是已知的。除了均值插补外，其他方法并没有考虑这个约束条件，没有将缺失数据本身的整体联系考虑进去，而均值插补对于较大缺失区间的填补效果并不理想。目前，也有一些研究是将能耗进行分解，SH Park 等人通过建立模型，将能耗的变化分解为三个因素：结构变化、能耗强度和输出水平[21]。欧育辉等人将总能耗增长分解为生产效应、结构效应和强度效应[22]。还有不少关于建筑中能耗拆分的研究[23][24]，将建筑能耗拆分为空调用电、照明用电等分项数据。但是这些方法，都是将能耗分解为固定几个分项，并不符合缺失区间大小不固定的场景。因此，考虑对其他算法进行改进。本文利用 KNN 的思想，提出在满足缺失数据区间和，基于连续等式约束优化的缺失数据填补方法，并将其运用到复旦大学节能监管平台中。

1.3 论文的研究内容

针对上述能耗监管平台存在缺失数据的问题，以及目前缺失数据填补方法存在的不足，本文提出基于约束优化的缺失能耗数据填补方法，通过与均值插补、回归插补、线性插补和三次样条插值填补进行对比，使用 MAE、MRE 和 MSE 作为评估指标，验证该方法有更好的填补效果。与此同时，针对复旦大学节能监管平台的数据特点和不同的缺失情况，本文制定了相应的缺失能耗数据填补策略，将基于约束优化的缺失能耗数据填补方法应用到该平台。本文的研究内容主要有以下三点：

1. 提出了基于约束优化的缺失能耗数据填补方法。该方法利用 KNN 的基本

思想，以 DTW 作为相似性度量，然后将该问题转化为连续等式约束优化问题，最终利用拉格朗日乘数法，将缺失区间内总能耗，分配到各个缺失点，完成填补。

2. 基于复旦大学节能监管平台的能耗数据，实现了基于约束优化的缺失能耗数据填补方法，做了一系列对比实验，与现存填补方法进行对比，验证了该方法具有更好的填补效果。

3. 提出了面向复旦大学节能监管平台的缺失能耗数据填补策略。根据复旦大学节能监管平台的具体数据存储格式和特点，制定特定的定时填补策略，开发成一个缺失能耗数据填补工具。

1.4 论文的组织与结构

本文总共为六章，每一章的概要内容如下：

第一章介绍了缺失能耗数据填补的研究背景和意义，分析国内外研究现状，总结现有缺失数据填补方法的缺陷，阐述本文研究内容，以及梳理论文结构。

第二章介绍相关原理和技术。详细介绍 K 近邻填补算法，动态时间规整距离，连续等式优化问题和拉格朗日乘数法。

第三章详细介绍基于约束优化的缺失数据填补算法，设计相关对比实验，将该算法与均值填补、回归填补和三次样条插值填补进行对比，验证该算法的有效性和精确性。

第四章详细分析了复旦大学节能监管平台的数据特点，总结缺失能耗数据的缺失情况，以及拓扑的变更对填补数据的影响。在此基础上制定了缺失能耗数据的填补策略，并对填补系统进行了系统设计。

第五章详细介绍了缺失能耗数据填补工具的系统实现，包括数据存储层、缓存层和定时器层。详细介绍了缺失区间的检测、预处理数据的缓存和缺失数据的填补的实现。

第六章总结本文的主要内容，并对研究内容的下一步改进进行思考，为未来的进一步工作提出展望。

第二章 相关原理与技术

2.1 K 近邻填补算法

K 最近邻算法最初由 Cover 和 Hart 于 1968 年提出[25]，是一种简单且易于快速实现的一种分类算法，其核心思想是在特征空间中寻找一个样本的 K 个最近邻样本，如果这 K 个最近邻大多数样本属于某一个类别，那么该样本也属于该类别。

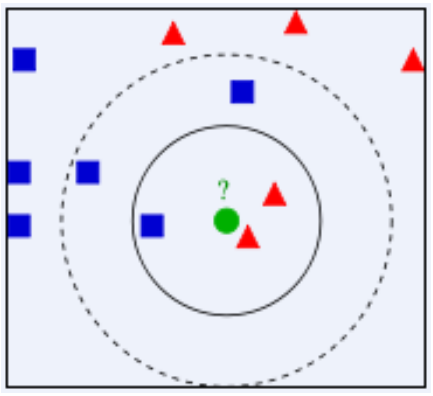


图 2.1 KNN 算法决策过程[26]

如图 2.1 所示，当 K=3 时，绿色圆点与红色三角形属于同一类别；当 K=5 时，绿色圆点与蓝色正方形属于同一类别。

利用这个思想，可计算存在缺失区间的时间序列的 K 近邻，得到相似集合，然后根据相似集合填补时间序列的缺失区间。填补的总体思路如图 2.2 所示。

第一步，选择 K 值，计算缺失时间序列与数据集中其他时间序列的相似性。目前时间序列的相似性度量主要有两种，一是明可夫斯基（Minkowski）距离，二是动态时间规整（Dynamic Time Warping）距离[27]。

第二步，根据与缺失时间序列的相似度，将数据集中的时间序列按照相似度从大到小排序，选取前 K 条时间序列，作为填补的参考集合。

第三步，根据选取的前 K 条时间序列填补缺失区间的各个数据项，一般可以选取前 K 条时间序列对应数据项的均值，或者根据不同权重进行填补。

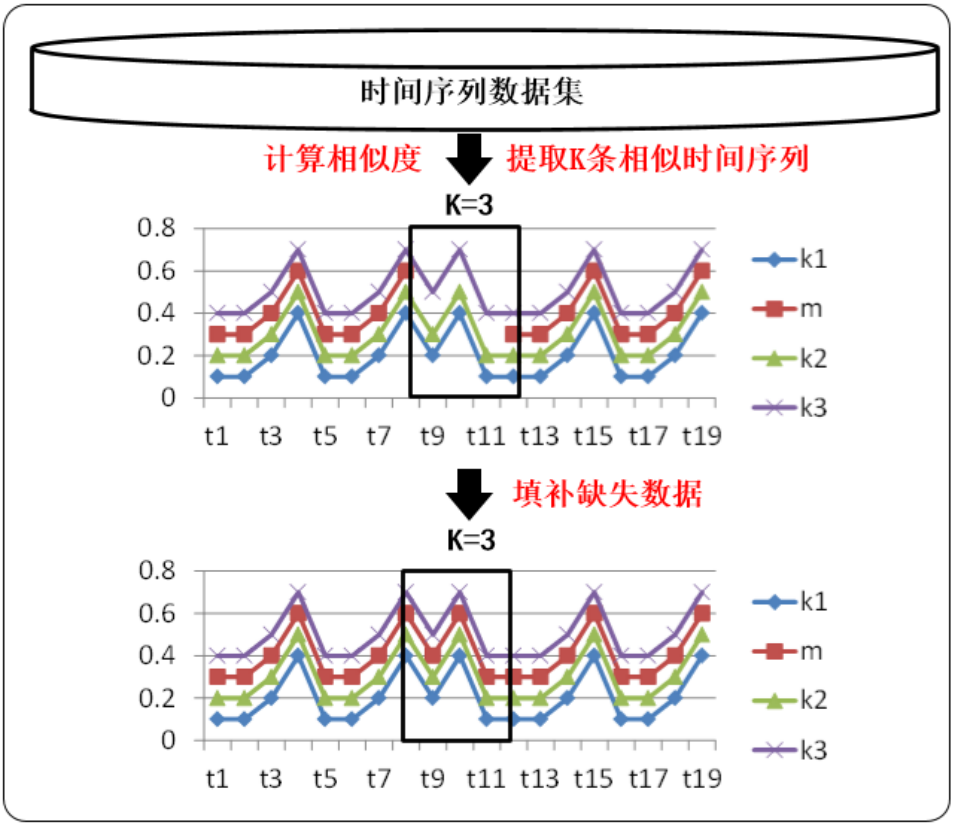


图 2.2 基于 KNN 的时间序列填补过程

2.2 动态时间规整距离

Dynamic Time Warping（DTW）动态时间规整是一种衡量两条时间序列相似性的算法，源自语言识别领域，主要用来识别两段录音是否表示相同的单词，由 Berndt 和 Clifford 引入数据挖掘领域[28]。与其他时间序列相似性度量相比，DTW 的优势在于，基于动态规划的思想，解决了不等长时间序列间的相似性度量问题。DTW 通过将时间序列进行延长和拉伸，来计算两条时间序列的相似度，如图 2.3 所示：

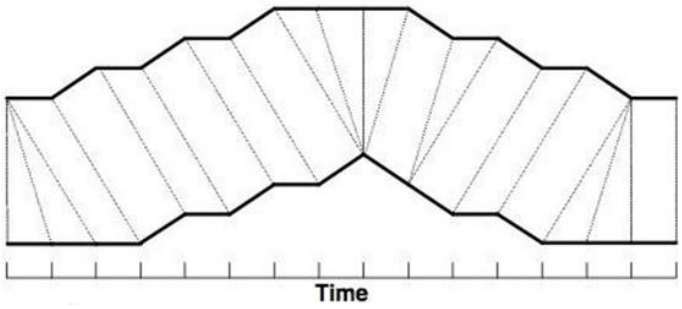


图 2.3 两个时间序列的规整[27]

图中实线表示两条时间序列，虚线表示时间序列间相似的点，这些相似的点

的距离之和，称为规整路径距离。

设时间序列 $X =< x_1, x_2, ..., x_n >$ 和 $Y =< y_1, y_2, ..., y_m >$，那么 X 和 Y 的规整路径距离 $D(X,Y)$ 定义为：

$$D(X,Y) = d(x_n, y_m) + \min \begin{cases} D(X, Y[1:m-1]) \\ D(X[1:n-1], Y) \\ D(X[1:n-1], Y[2:m-1]) \end{cases}$$

其中，$d(x_i, x_j)$ 表示 x_i 和 x_j 之间的距离，一般采用欧式距离。为了实现时间序列的线性缩放，DTW 在实践中采用动态规划的方法求解，具体步骤如下：

第一步，为了对齐两个时间序列，构造一个 n*m 的代价矩阵，矩阵中 (i, j) 处的值为 $d(x_i, x_j)$。

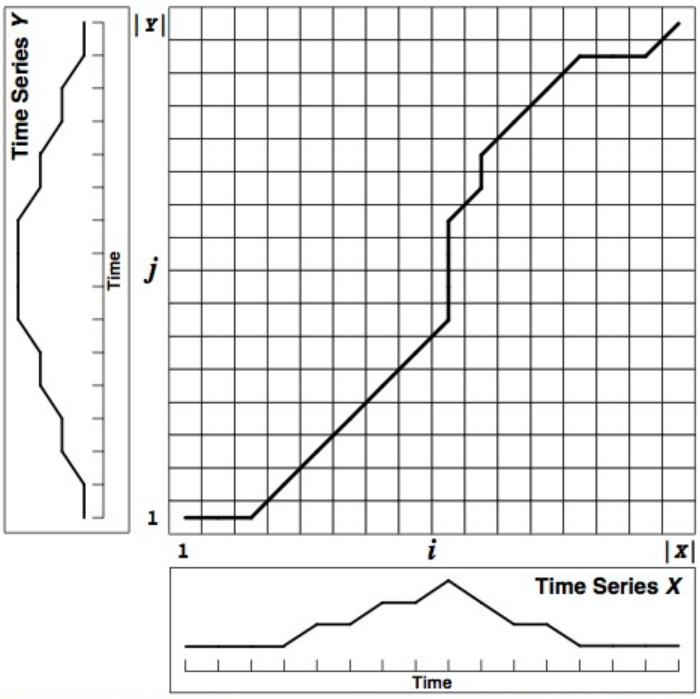


图 2.4 代价矩阵和最小规整路径[27]

第二步，寻找一条通过矩阵若干点的路径，称为规整路径，表示为：

$$W = w_1, w_2, ..., w_k, ...w_K \quad \max(m.n) \le K < m+n$$

其中 $w_k = (i, j)_k = d(x_i, x_j)$，规整代价最小的规整路径上的点值之和即为规整路径距离。关于规整路径，需要满足以下约束：

1）边界条件：

$$w_1 = (1,1)$$

$$w_K = (n, m)$$

时间序列的各个部分的先后次序是不可改变的，所选的路径的起点必须是左下角，终点为右上角。

2）连续性：

若 $w_{k-1} = (a', b')$，那么规整路径的下一个点 $w_k = (a, b)$ 需要满足：

$$(a - a') \leq 1$$

$$(b - b') \leq 1$$

在进行时间序列匹配过程中，不能跨越某个点匹配，这样保证两个时间序列的每个坐标都能在 W 中出现。

3）单调性：

若 $w_{k-1} = (a', b')$，那么规整路径的下一个点 $w_k = (a, b)$ 需要满足：

$$0 \leq (a - a')$$

$$0 \leq (b - b')$$

在进行时间序列匹配过程中，必须随着时间单调进行，不能往回匹配，保证了图 2.3 中的虚线不会相交。

满足上述约束的路径有指数个，此时，使用动态规划算法找出规整代价最小的那条规整路径，最终得到规整距离。算法 2.1 具体描述了 DTW 的求解过程，输入两个时间序列 $X = (x_1, ..., x_n)$和$Y = (y_1, ..., y_m)$，输出时间序列 X 和 Y 的规整距离 $D(X, Y)$。

算法 2.1：计算两个时间序列的规整距离

输入：

两个时间序列：$X = (x_1, ..., x_n)$，$Y = (y_1, ..., y_m)$

输出：

两个时间序列的规整距离：$D(X, Y)$

算法过程：

DTW(X, Y)

1. Dim Mat As Array //定义二维数组 Mat，保存代价矩阵

```
2.  Dim Dis As Array //定义二维数组 Dis，Dis(i,j)表示 X[1:i] 和 Y[1:j] 的规整距离
3.  n ← X.length, m ← Y.length
4.  for i ← 1 to n
5.      for j ← 1 to m
6.          Mat(i,j) ← (x_i – y_j)^2
7.      end
8.  end
9.  for i ← 1 to n
10.     for j ← 1 to m
11.         Dis(i,j) ← MAX_VALUE //初始化 Dis(i,j)为一个很大的数值
12.         if i-1 ≥ 1
13.             Dis(i,j) ← min(Dis(i,j) , Dis(i-1,j) + Mat(i,j))
14.         end
15.         if j-1 ≥ 1
16.             Dis(i,j) ← min(Dis(i,j) , Dis(i,j-1) + Mat(i,j))
17.         end
18.         if i-1 ≥ 1 and j-1 ≥ 1
19.             Dis(i,j) ← min(Dis(i,j) , Dis(i-1,j-1) + Mat(i,j))
20.         end
21.         if i == 1 and j == 1
22.             Dis(i,j) ← min(Dis(i,j) , Mat(i,j))
23.         end
24.     end
25. end
26. return Dis(n,m)
```

2.3 等式约束优化问题

为了将区间总能耗更好地分配到每个缺失点，可以将其看作等式约束优化问题。通常约束优化问题被定义为[29]：

定义 2.3 约束优化问题：

最小化 $f(\mathbf{x}), \mathbf{x} = (x_1, ..., x_n)$

满足 $g_m(\mathbf{x}) \le 0, m = 1, ..., n_g$

$h_m(\mathbf{x}) = 0, m = n_g + 1, ..., n_g + n_h$

$x_j \in dom(x_j)$

式中 n_g 和 n_h 分别是不等式约束与等式约束的个数，$dom(x_j)$ 是变量 x_j 的取值范围。当不等式个数为 0 时，则该问题为等式约束优化问题。

线性等式约束 $\mathbf{Ax} = \mathbf{b}$ 是约束优化问题的一个特例，定义如下：

定义 2.4 带线性等式约束的约束优化：

最小化 $f(\mathbf{x}), \mathbf{x} = (x_1, ..., x_n)$

满足 $\mathbf{Ax} = \mathbf{b}$

式中 $\mathbf{A} \in \mathbb{R}^{n_h \times n_x}$，并且 $\mathbf{b} \in \mathbb{R}^{n_h}$。

2.4 拉格朗日乘数法

等式约束优化问题可以通过定义拉格朗日函数，将其转化为非约束优化问题[29]。例如定义 2.3 的标准约束优化问题，可以将式中的约束引入到目标函数 f 中，则拉格朗日函数为：

$$L(\mathbf{x}, \lambda_h) = f(\mathbf{x}) + \sum_{m=1}^{n_h} \lambda_{hm} h_m(\mathbf{x})$$

式中 $\lambda_h \in \mathbb{R}^{n_h}$ 为等式约束的权重，称为拉格朗日乘数。令 $L(\mathbf{x}, \lambda_h)$ 对 $\mathbf{x}$ 和 λ_h 的一阶偏导数等于零，即：

$$\begin{cases} L'_{x_i} = f'_{x_i}(\mathbf{x}) + \sum_{m=1}^{n_h} \lambda_{hm} h'_{mx_i}(\mathbf{x}) = 0, x_i \in \mathbf{x} \\ L'_{\lambda_{hm}} = \sum_{m=1}^{n_h} h_m(\mathbf{x}) = 0, \lambda_{hm} \in \lambda_h \end{cases}$$

由上述方程组求得 $\mathbf{x}$，就是 $f(\mathbf{x})$ 在约束 $\sum_{m=1}^{n_h} \lambda_{hm} h_m(\mathbf{x})$ 下的最优解。

2.5 小结

本章节详细介绍了 K 近邻填补的算法步骤，主要有三个步骤：计算时间序列相似度、提取 K 个最近邻时间序列和计算填补值。介绍了动态时间规整的算法步骤，使用伪代码详细说明了如何用动态规划算法计算动态时间规整距离。介绍了等式约束优化问题的定义，以及如何使用拉格朗日乘数法求解等式约束优化问题。

第三章 基于约束优化的缺失数据填补算法

复旦大学节能监管平台采用电表读数相减法来计算区间能耗值，因此缺失能耗区间的能耗总和是已知的。本文利用 KNN 的思想，提出在满足缺失数据区间和，基于等式约束优化的缺失数据填补算法，并将其运用到复旦大学节能监管平台中。

3.1 缺失数据填补算法详细步骤

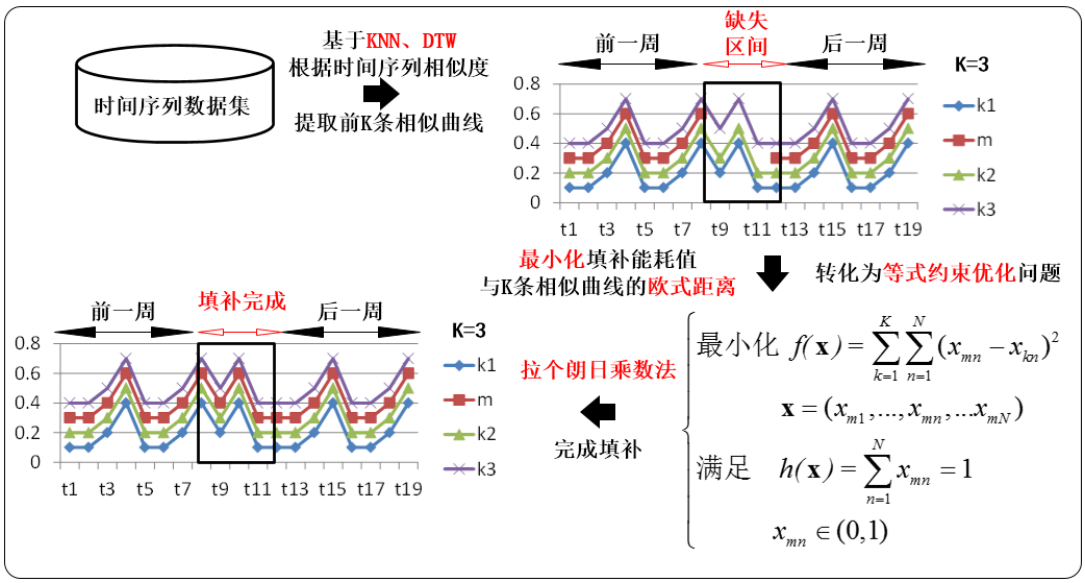


图 3.1 填补算法流程

如图 3.1 所示，填补算法的详细流程分为以下几个步骤：

第一步，提取 K 个相似时间序列。采用 DTW 时间序列相似性度量，计算出与缺失时间序列前一周和后一周，最相似的 K 个时间序列，时间粒度为小时，即每小时一个数据点。由于在计算相似性时，前一周和后一周同样可能存在缺失数据，导致时间序列不等长，因此选择 DTW 时间序列相似性度量，避免对噪声数据敏感。

第二步，计算数据点占区间总和比例。计算 K 个时间序列中，每个序列缺失区间所对应的各个数据点，占区间能耗总和的比例。用 x_{kn} 表示第 k 个时间序列，缺失区间第 n 个数据点占区间能耗总和的比例。

第三步，将问题转化为等式约束优化问题。用 x_{mn} 表示缺失时间序列缺失区间中，第 n 个待求的缺失数据点占区间数据总和的比例。通过最小化缺失时间序

列中，各个缺失数据点比例与 K 个时间序列对应的数据点比例的欧式距离，同时满足缺失数据点比例总和为 1，来求得 N 各待求的缺失数据点占区间能耗总和的比例，如下式所示：

$$\text{最小化} \quad f(\mathbf{x})=\sum_{k=1}^{K}\sum_{n=1}^{N}(x_{mn}-x_{kn})^2,\mathbf{x}=(x_{m1},...,x_{mn},...x_{mN}) \tag{3.1}$$

$$\text{满足} \quad h(\mathbf{x})=\sum_{n=1}^{N}x_{mn}-1=0 \tag{3.2}$$

$$x_{mn}\in(0,1) \tag{3.3}$$

式中 N 表示缺失区间数据点的个数。此时，便将问题转化为典型的等式约束优化问题，可通过拉格朗日乘数法求解。

第四步，利用拉格朗日乘数法，求解等式约束优化问题。定义拉格朗日函数，通过加权，将（3.2）式中的约束引入目标函数（3.1）中：

$$L(\mathbf{x})=f(\mathbf{x})+\lambda h(\mathbf{x})=\sum_{k=1}^{K}\sum_{n=1}^{N}(x_{mn}-x_{kn})^2+\lambda(\sum_{n=1}^{N}x_{mn}-1) \tag{3.4}$$

对 $\mathbf{x}$ 和 λ 求偏导，可得：

$$L'_{x_{mn}}=2\sum_{k=1}^{K}(x_{mn}-x_{kn})+\lambda=2Kx_{mn}-2\sum_{k=1}^{K}x_{kn}+\lambda=0 \tag{3.5}$$

$$L'_{\lambda}=\sum_{n=1}^{N}x_{mn}-1=0 \tag{3.6}$$

其中 $n\in[1,N]$，根据式（3.5）可得：

$$x_{mn}=\frac{\sum_{k=1}^{K}x_{kn}-\frac{\lambda}{2}}{K} \tag{3.7}$$

将式（3.7）带入式（3.6）中，可得：

$$L'_{\lambda}=\sum_{n=1}^{N}\frac{\sum_{k=1}^{K}x_{kn}-\frac{\lambda}{2}}{K}-1=0 \tag{3.8}$$

化简式（3.8），可求得：

$$\lambda = \frac{2\sum_{n=1}^{N}\sum_{k=1}^{K} x_{kn} - 2K}{N} \tag{3.9}$$

最后，将式（3.9）带回式（3.7）中，即可求得 x_{mn}。

算法 3.1 描述了对缺失时间序列 $seq=(X,...,X)$ 进行填补的过程，算法的输入为缺失时间序列 $seq=(X,...,X)$，缺失区间和 sum，所有时间序列集合 $seqs$，最近邻个数 K，输出填补后的时间序列 $seq'=(X,...,x_{m1},...,x_{mN},...,X)$。

算法 3.1：基于约束优化的缺失数据填补算法

输入：

一个缺失时间序列： $seq=(X,...,X)$；

缺失区间和： sum；

所有时间序列集合： $seqs$；

最近邻个数：K；

输出：

填补后的时间序列： $seq'=(X,...,x_{m1},...,x_{mN},...,X)$

算法过程：

Interpolation(seq,sum,seqs,K)

1. *sims ← {} , simsK ← {}*
2. *seqCompressLeft ← Compress(seq[m1-24*7:m1-1]) //压缩缺失区间前一周数据为百分比*
3. *seqCompressRight ← Compress(seq[mN+1:mN+24*7]) //压缩缺失区间后一周数据为百分比*
4. *for i ← 0 to seqs.size*
5. *seqSCompressLeft ← Compress(seqs[i][m1-24*7:m1-1])*
6. *seqSCompressRight ← Compress(seqs[i][mN+1:mN+24*7])*

```
seqSCompressMissing ← Compress(seqs[i][m1:mN]) //压缩缺失区间内数据为百分比
dtwSiml ← DTW(seqCompressLeft，seqSCompressLeft) //计算缺失区间前一周序列相似度
dtwSimr ← DTW(seqCompressRight , seqSCompressRight) //计算缺失区间后一周序列相似度
dtwSim ← (dtwSiml + dtwSimr)/2
sims.put(seqSCompressMissing , dtwSim)
end
sims.sort() //根据 dtwSim 降序排序
simsK ← sims.sub(0 , K-1)
temp ← 0
for n ← 0 to N-1
    for k ← 0 to K-1
        temp ← temp + simsK[k][n]
    end
end
λ ← (2*temp-2*K)/N
for n ← 0 to N-1
    temp ← 0
    for k ← 0 to K-1
        temp ← temp + simsK[k][n]
    end
    p ← (temp –λ/2)/K
    seq.put(p*sum)
end
```

30. return seq

3.2 对比实验

针对本文提出的基于约束优化的缺失数据填补算法，设计了相关实验来验证该算法的有效性和精确性。实验数据来源于复旦大学绿色校园节能监管平台能耗数据。实验内容主要将本文提出的填补算法与均值填补、回归填补、三次样条插值填补进行对比，通过填补结果的精确率来评估填补的质量。最后分析本文的填补算法的实验结果。

3.2.1 相关算法

为了使最终的填补结果满足区间能耗总和，在回归填补和三次样条插值填补中，将能耗总和按照各个数据点填补值所占填补值总和的比例，分摊到各个数据点。

3.2.1.1 均值填补

均值填补是一种简便快速的缺失数据填补方法[30]。当缺失数据为数值类型时，根据数据集中其他对象取值的平均值作为填补值；当缺失数据为非数值类型时，则选取数据集中的众数作为填补值。

在实验过程中，均值填补主要分为两步：

第一步，压缩缺失区间数据。将在缺失区间未发生缺失的所有时间序列，对应缺失区间内的数据压缩成百分比。

第二步，填补数据。计算压缩数据的平均值，得到填补缺失区间的百分比，再根据缺失区间总和，计算得到填补值。

3.2.1.2 回归填补

回归填补[31][32]是基于完整数据集，将缺失属性作为因变量，相关属性作为自变量，根据它们之间的关系建立回归模型。多元线性回归填补步骤如下：

第一步，选择自变量建立回归模型。设因变量为 y，表示缺失属性，选择的自变量为 $x_1, x_2, ..., x_n$，表示缺失属性的相关属性，则多元线性回归模型为：

$$y=\beta_0+\beta_1x_1+\beta_2x_2+...+\beta_nx_n+\varepsilon$$

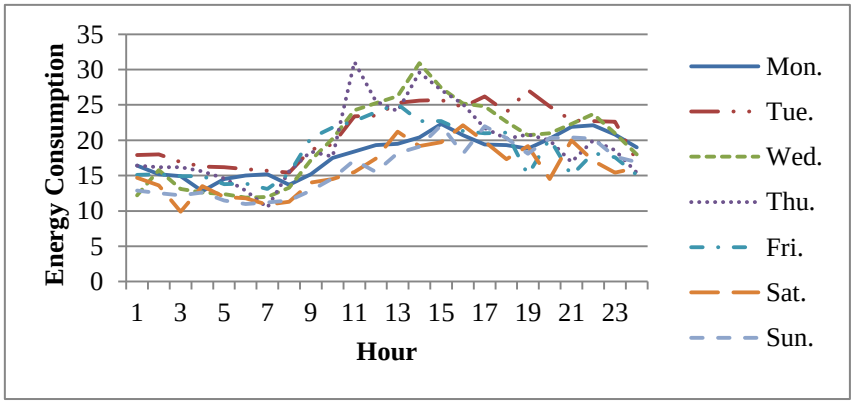
其中，β 为待求参数，ε 服从正态分布等经典假定。

第二步，计算 β 的估计值。利用训练数据，使用最小二乘法求得 β 的估计值 $\hat{\beta}_0,\hat{\beta}_1,\hat{\beta}_2,...,\hat{\beta}_n$，得到多元线性回归方程：

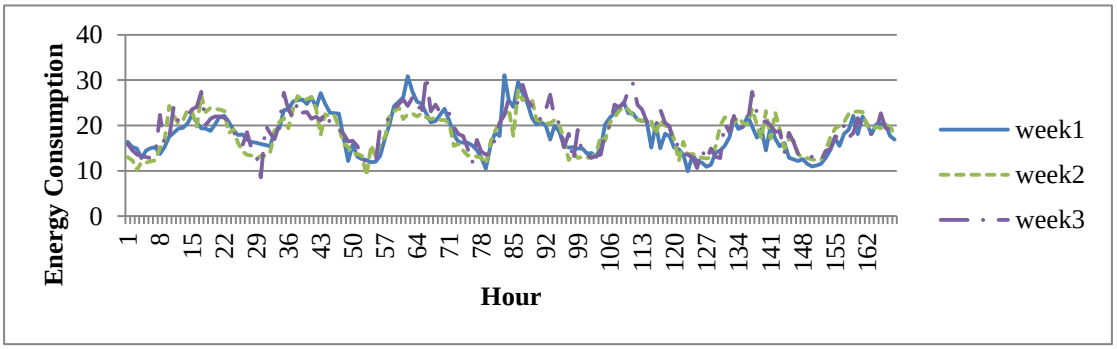
$$\hat{y}=\hat{\beta}_0+\hat{\beta}_1x_1+\hat{\beta}_2x_2+...+\hat{\beta}_nx_n \tag{3.10}$$

第三步，将自变量的值带入式 3.10 中，得到填补值。

根据复旦大学节能监管平台的数据特点，如下图所示，为复旦大学恒隆物理楼 2016 年 4 月 4 日到 2016 年 4 月 10 日 7 天的能耗数据，以及 2016 年 4 月 4 日到 2016 年 4 月 24 日 3 周的能耗数据：



（a） 恒隆物理楼 7 天能耗数据



(b) 恒隆物理楼 3 周能耗数据

由上图可以发现，能耗数据以天和周为规律明显，因此，选取缺失数据点前 24 小时、前一周、后 24 小时和后一周的数据作为自变量，分别用 x_{preday}、$x_{preweek}$、$x_{postday}$ 和 $x_{postweek}$ 表示，则回归方程为：

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_{preday} + \hat{\beta}_2 x_{preweek} + \hat{\beta}_3 x_{postday} + \hat{\beta}_4 x_{postweek}$$

3.2.1.3 三次样条插值填补

三次样条插值是一种数值方法，Baltazar 等人完整描述了如何利用这种方法进行填补缺失数据[11]。假设曲线经过点 $(x_0, y_0),...,(x_n, y_n)$，其中 $a = x_0 < x_1 < \cdots < x_n = b$。样条曲线 $S(x)$ 是一个分段函数，给定 n 个区间，三次样条函数需满足一下条件：

1）当处在分段区间 $[x_i, x_{i+1}](i = 0,1,...,n-1)$ 时，$S(x) = S_i(x)$，为一个三次多项式；

2）$S(x_i) = y_i, (i = 0,1,...,n-1)$；

3）$S(x)$，$S^{'}(x)$，$S^{''}(x)$ 在区间 $[a,b]$ 上是连续的。

n 个分段三次多项式写作：

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3，\ i = 0,1,...,n-1$$

其中 a_i，b_i，c_i，d_i 为待求参数。

3.2.2 实验数据

实验数据来自于复旦大学节能监管平台，包含 483 个一般用电、空调用电和照明用电测点，从 2016 年 1 月 1 日到 2016 年 12 月 31 日的每小时能耗数据，以及这些测点的详细信息。选取恒隆物理楼空调用电作为测试测点，并生成 3 种不同大小的缺失区间，分别为 24 小时、4 天和 1 周。

由于在真实场景中，数据的缺失并不是随机的，因此，借鉴[14]构造缺失数据的方法，在 2 月份、3 月份、8 月份、9 月份中寻找真实缺失区间，然后分别映射到 4 月份、5 月份、10 月份、11 月份中构造缺失区间，具体做法如图 3.2 所示。2 月份、3 月份、8 月份和 9 月份中的红色 X 表示数据缺失，对应到 4 月份、5 月份、10 月份和 11 月份的数据，例如 5 月份中的 9 和 12，作为测试数据。

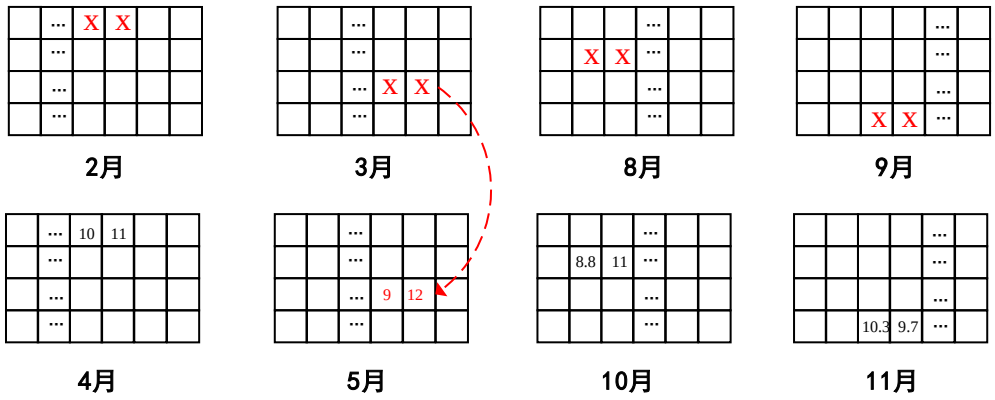


图 3.2 缺失区间的选取

如图 3.2 所示，3 月份存在缺失区间，那么 5 月份对应的区间的数据作为测试数据，其真实值用于评估填补精度。用相同的方法，选取 4 月份、10 月份和 11 月份的测试数据。在所有缺失区间中，选取连续缺失 24 小时、4 天和 1 周的区间，作为最后的测试数据。

3.2.3 评估指标

本文采用平均绝对误差（MAE）、平均相对误差（MRE）和均方误差（MSE）作为算法精确性评估指标，

$$MAE=\frac{\sum_{i=1}^{n}\left|x_i-\hat{x}_i\right|}{n},\quad MRE=\frac{\sum_{i=1}^{n}\left|x_i-\hat{x}_i\right|}{\sum_{i=1}^{n}x_i},\quad MSE=\frac{\sum_{i=1}^{n}\left(x_i-\hat{x}_i\right)^2}{n},$$

其中，$\hat{x}_i$ 表示填补值，x_i 表示真实值，n 表示缺失区间大小。

3.2.4 实验结果

使用基于约束优化的缺失数据填补算法，填补区间大小为 24 小时、4 天和 1 周的缺失数据，并计算平均 MAE、MRE 和 MSE。如图 3.3 所示，当 K=12 时，基于约束优化的缺失数据填补算法的误差最小，因此选取 K 为 12 与其他填补算法进行比较。

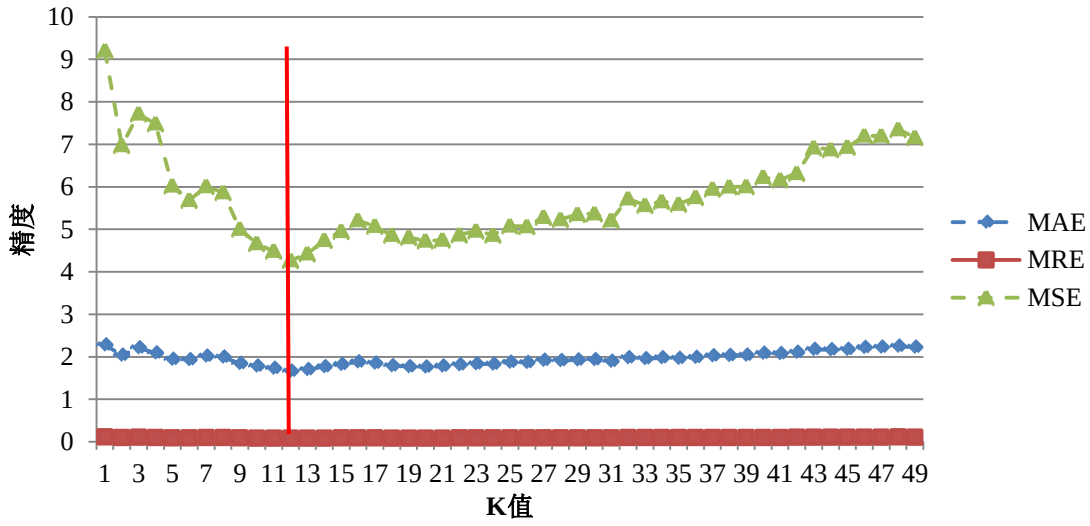


图 3.3 基于约束优化填补的精度随 K 的取值的变化

如图 3.4、3.5 和 3.6 所示，分别为缺失区间为 24 小时、4 天和 1 周的填补结果对比图，其中红色实线表示真实能耗曲线，其他虚线分别为基于约束优化填补、回归填补、三次样条填补和均值填补的曲线。可以明显看出基于约束优化的填补算法，更能还原时间序列的趋势，而三次样条插值的效果最差。

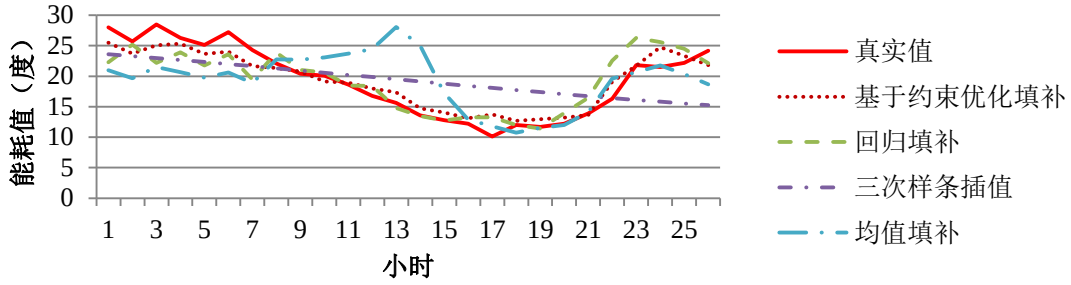


图 3.4 24 小时填补结果

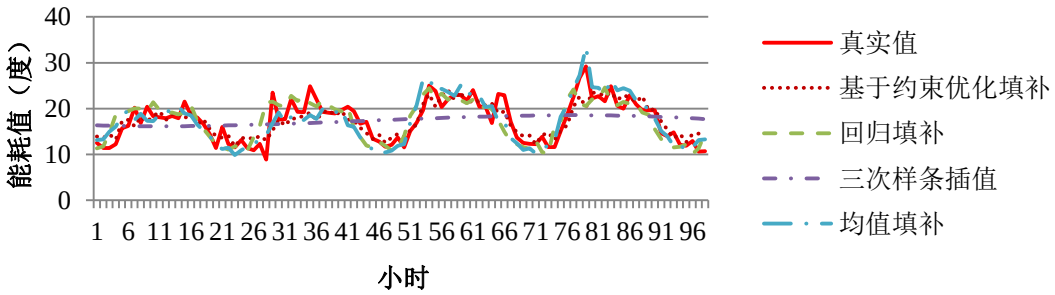


图 3.5 4 天填补结果

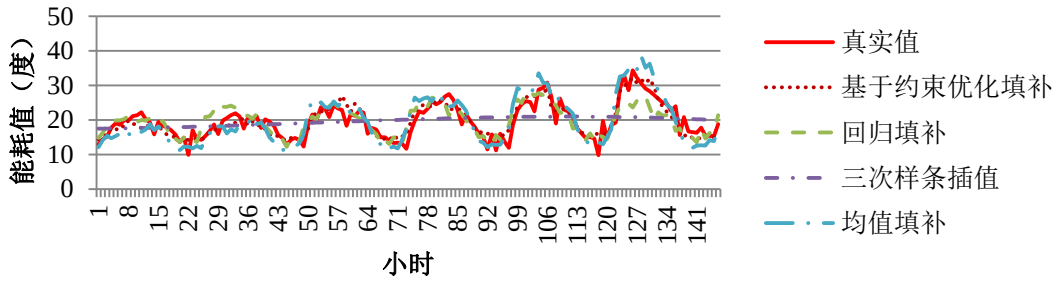


图 3.6 一周填补结果

表 3.1 为四种填补算法的填补精度对比，可以看出，基于约束优化的填补算法对三种不同大小的缺失区间的填补效果，都是最好的。

表 3.1 四种填补算法填补精度对比

算法	24 小时			4 天			一周		
	MAE	MRE	MSE	MAE	MRE	MSE	MAE	MRE	MSE
均值填补	4.088	0.211	27.78	2.118	0.126	7.023	2.772	0.141	11.58
三次样条插值	4.190	0.216	22.85	3.756	0.337	18.77	3.938	0.200	22.71
回归填补	2.316	0.120	9.174	2.203	0.126	8.883	2.307	0.117	9.403
基于约束优化填补	**1.583**	**0.082**	**3.578**	**1.822**	**0.105**	**5.862**	**1.743**	**0.089**	**4.955**

3.3 小结

本章节详细描述基于约束优化的缺失数据填补的算法步骤，主要分为四个步骤：1）提取 K 个相似时间序列；2）计算数据点占区间总和比例；3）将问题转化为等式约束优化问题；4）利用拉格朗日乘数法，求解等式约束优化问题。并且设计了对比实验，与均值填补、回归填补和三次样条插值填补进行比较，验证了该算法的有效性和精确性。

第四章 缺失能耗数据填补策略与系统设计

为了将基于约束优化的缺失数据填补算法运用到复旦大学节能监管平台，需要分析复旦大学节能监管平台数据特点，根据特定的数据缺失场景，制定相应的具体的缺失能耗数据填补策略，并对缺失数据填补工具进行系统设计。

4.1 分析复旦大学节能监管平台数据

4.1.1 测点概念介绍

测点，分为真实测点和虚拟测点。真实测点，是指直接由电表读取数值的测点；虚拟测点，是指数值通过其他测点的值相加而得的测点。在复旦大学节能监管平台中，测点的所属对象分为四个层级：房间、楼宇、区域和校区。

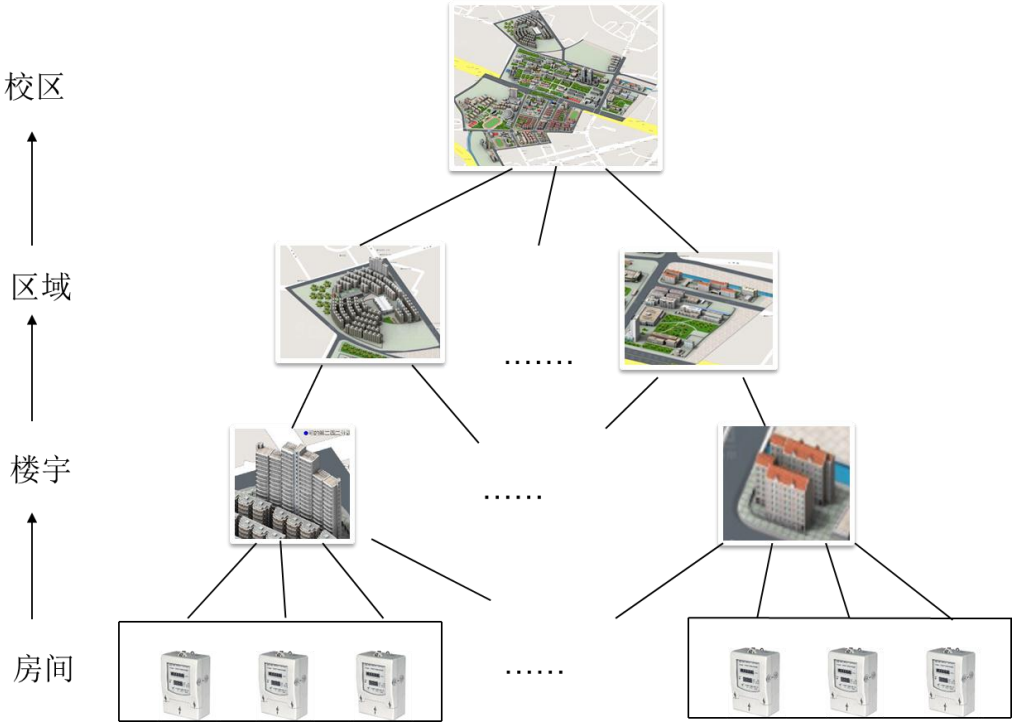


图 4.1 测点的所属对象层级划分

如图 4.1 所示，为测点的所属对象的主要层级划分。各个层级都有空调用电、照明用电和动力用电等分项用电的测点，例如光华楼有空调用电、照明用电、动力用电和电梯用电四个测点，这四个测点的所属对象是光华楼，属于楼宇层级。上一级测点的能耗值一般由下一级测点的能耗值相加而得，用于相加的下级测点称为该上级测点的子测点，上级测点称为父测点，真实测点没有子测点。

4.1.2 数据的采集

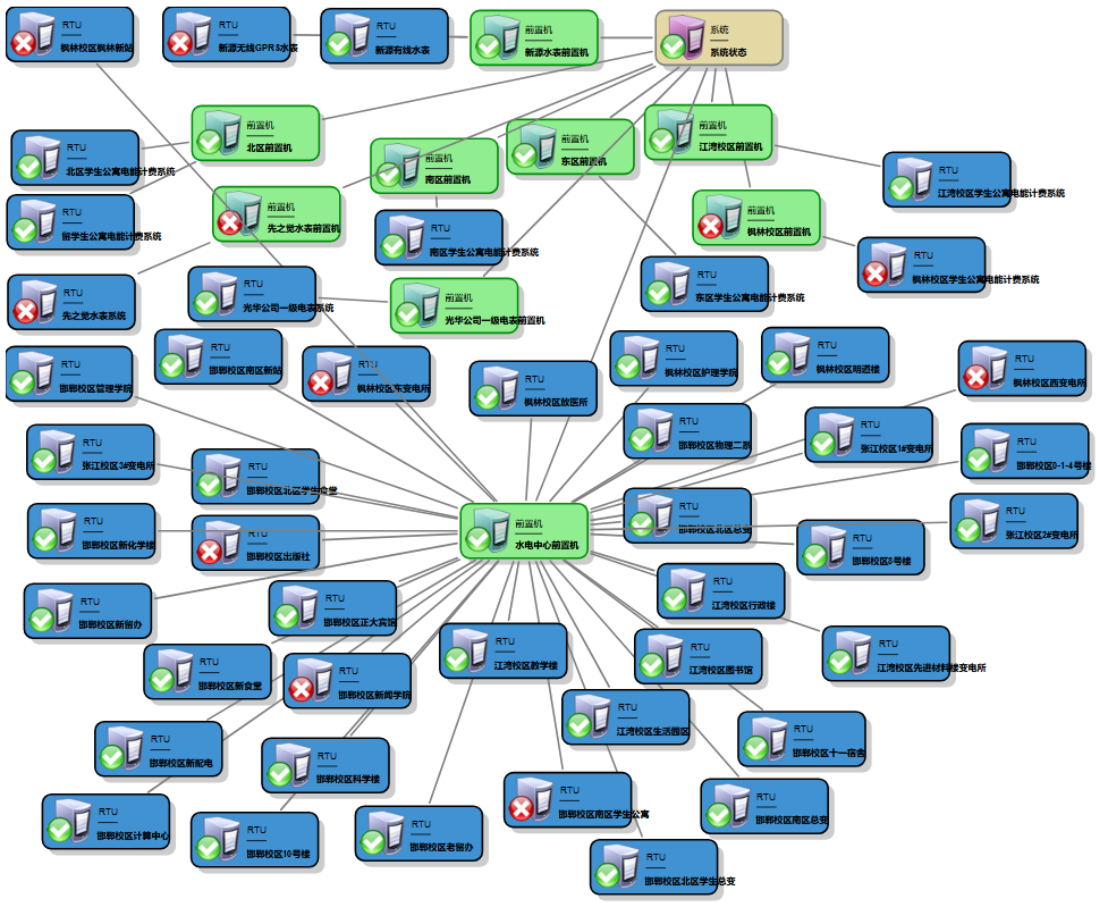


图 4.2 数据采集拓扑图

如图 4.2 所示，为复旦大学节能监管平台的数据采集拓扑图，蓝色图标代表 RTU，从电表读取数据；绿色图标代表前置机，从 RTU 采集能耗数据，然后将数据统一处理成固定格式，上传至系统数据库，即图中黄色图标。目前，平台中有 20000 多个真实测点，能耗数据的采集频率为 15 分钟一次，每小时大约有 9 万条记录写入数据库。由于能耗数据为电表的读数，因此这些能耗数据随着时间递增。

4.1.3 能耗值的计算

能耗值的计算主要涉及测点的能耗值计算和建筑的能耗值计算，如图 4.3 所示，红色和蓝色的圆代表不同能耗类型的测点（比如空调用电、照明用电等），一个对象（即校区、区域、楼宇和房间）可以拥有多个测点，测点 1 和测点 2 不属于任何对象，测点 1 统计的是楼 2 和楼 3 的总能耗，测点 2 统计的是楼 3 的能耗。

1）测点的能耗值。父测点的能耗值由下一层级子测点的能耗值相加而得。如果子测点为真实测点，那么子测点的能耗值为采集的电表读数；否则，该子测点的能耗值由再下一层级子测点的能耗值相加而得。

2）建筑的能耗值。建筑能耗值的计算分为三种，第一种是将某栋楼下的各个分项能耗的测点的能耗值相加，如楼 1；第二种是由单个测点直接统计，如楼 3；第三种是通过公式计算而得，如楼 2 是测点 1 的能耗值减去测点 2 的能耗值。由于平台中的能耗值都是电表的读数，因此，在统计某段时间内建筑所消耗的电量是，需要将该段时间区间的结束时间能耗值减去起始时间能耗值。

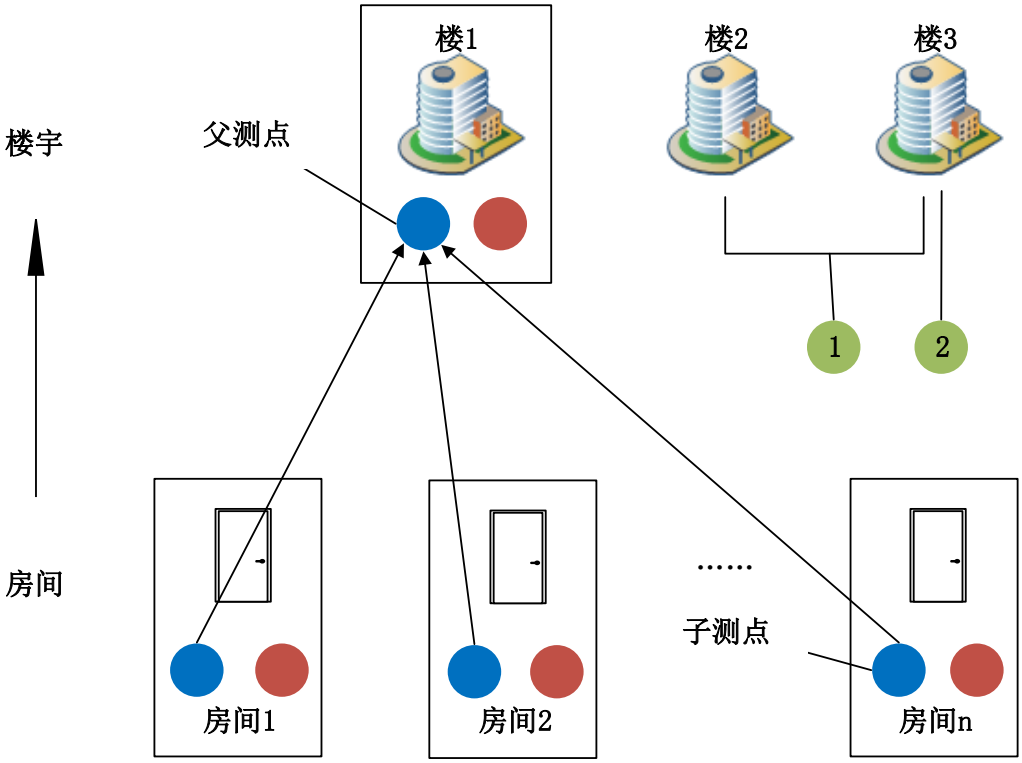


图 4.3 能耗值的计算

4.1.4 能耗数据缺失情况分类

由于复旦大学节能监管平台的测点拓扑结构和层级结构比较复杂，因此有必要深入分析和了解能耗数据缺失的特点，以便制定特定的缺失能耗数据填补策略，以下是对能耗数据缺失情况的分类。

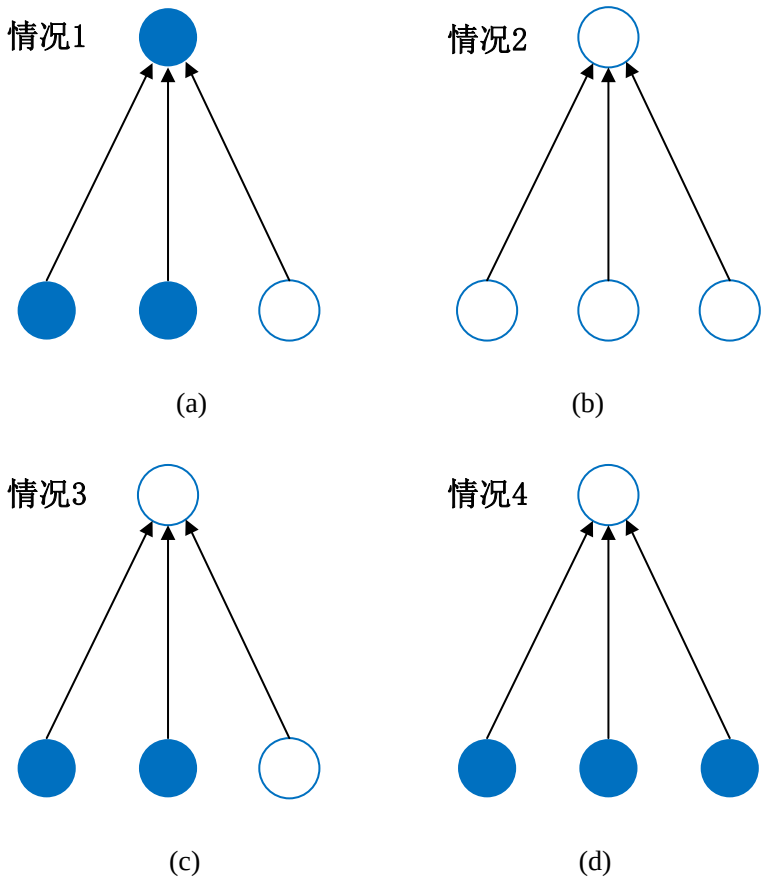


图 4.4 能耗数据缺失情况

如图 4.4 所示，实心的圆表示在指定时间内未发生数据缺失，空心的圆表示在指定时间内发生数据缺失。能耗数据的缺失情况分为两种：

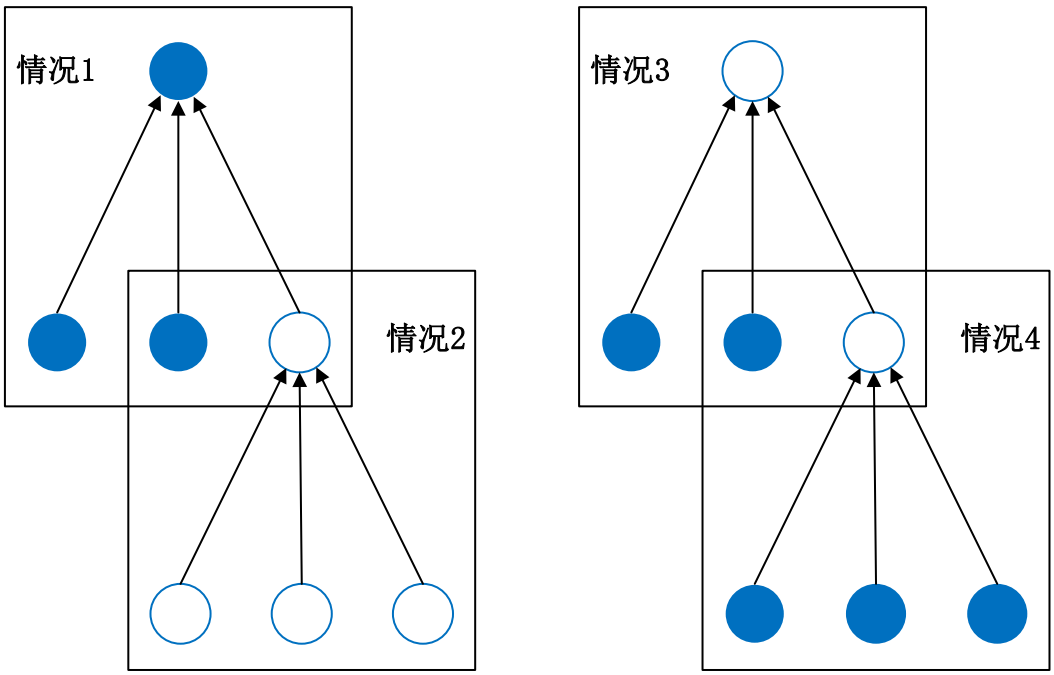
1）情况 1，部分子测点发生能耗数据缺失，父测点未发生能耗数据缺失，但是有数据遗漏；

2）情况 2，子测点全部发生能耗数据缺失，导致父测点发生能耗数据缺失；

3）情况 3，部分子测点发生能耗数据缺失，由于某种原因，父测点能耗值没有统计而缺失；

4）情况 4，子测点没有发生能耗数据缺失，由于某种原因，父测点能耗值没有统计而缺失。

由于测点的所属对象是多层级的结构，因此能耗数据的缺失通常是由各个情况的组合而成，如图 4.5 所示，图 4.5(a)为情况 1 和情况 2 的组合，图 4.5(b)为情况 3 和情况 4 的组合，还有更多组合，这里就不一一列举了。



(a) (b)

图 4.5 情况 1 和情况 2 的组合

根据数据缺失的时间跨度，可以分为短期缺失、中期缺失和长期缺失，分别为缺失 24 小时以内、24 小时到一周和一周以上。

4.1.5 测点拓扑变更

复旦大学节能监管平台在运行过程中，由于以下因素，经常出现测点拓扑结构的变更：

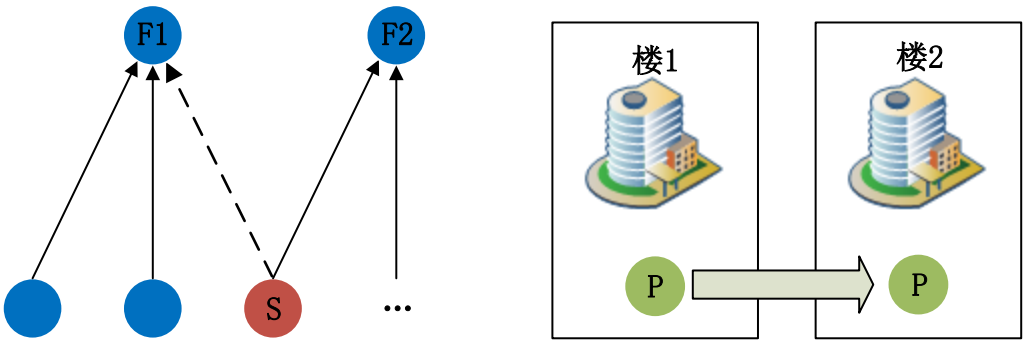
1）楼寓建筑的增加和改造；

2）电网线路的调整，电网拓扑结构的变更，导致测点信息的变更；

3）智能表具、采集器发生更换；

4）智能表具变更接入系统，如原来通过在水电中心的电能数据采集系统接入校园节能监管平台，改为通过校园变电站电力监控系统接入校园节能监管平台，对这些测点在监管平台中的对应关系进行重新配置。

如图 4.6 所示，为测点变更的例子，图 4.6(a)表示子测点 S 的父测点由测点 F1 变为 F2，图 4.6(b)表示测点 P 的所属对象由楼 1 变为楼 2。



(a) (b)

图 4.6 测点拓扑变更

测点拓扑的变更，会导致测点历史数据与当前实际情况不同。例如图 4.6(b)的拓扑变更的情况，当统计楼 2 的历史数据时，由于测点 P 的历史数据时属于楼 1，因此会导致楼 2 的统计数据不正确。

4.2 缺失能耗数据填补策略

为了保证填补数据的精确性，对于长期缺失，即缺失一周以上的数据，不进行填补。

4.2.1 针对不同缺失情况

1）对于情况 1，对子测点进行填补，父测点的统计遗漏，在查询时补充叠加。

2）对于情况 2，先对子测点进行填补，父测点的填补值为子测点的填补值之和。

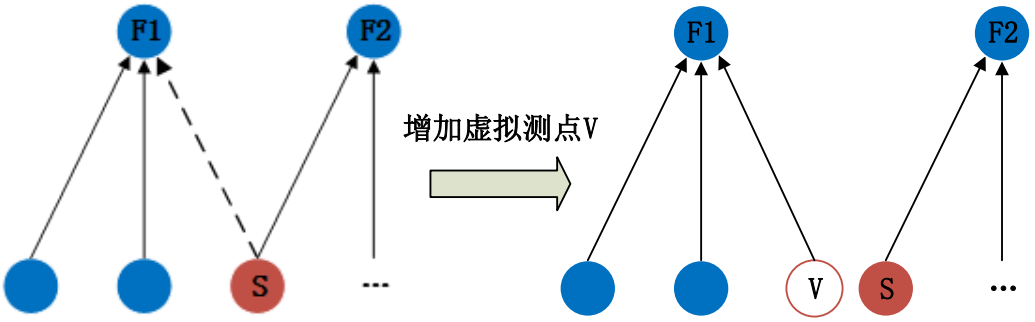
3）对于情况 3，先对子测点进行填补，父测点的填补值为无缺失子测点的真实值和缺失子测点的填补值之和。

4）对于情况 4，直接将所有子测点的能耗值相加，得到父测点的填补值。因子测点并未发生能耗数据缺失，因此此时的填补值，实际上是真实值。

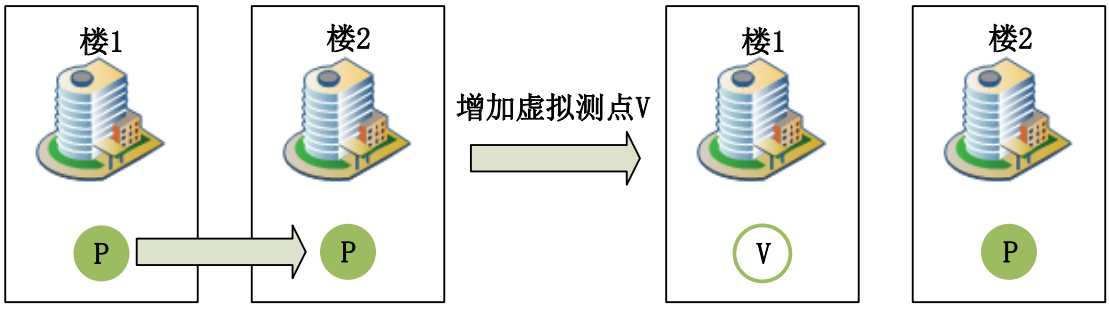
综上所述，所有的数据缺失情况，都需要从子测点开始填补。因此采用递归填补策略，即在填补当前测点时，先对子测点进行填补，如果子测点也存在缺失子测点，那么先对子测点的子测点进行填补，如此递归下去，知道不存在子测点。然后，在返回上一层递归时，将子测点求和，得到父测点的填补值。填补策略的详细过程将在系统实现中描述。

4.2.2 针对拓扑变更

拓扑的变更主要会导致历史数据的错乱，因此关键在于如何处理已经填补的能耗数据。本文采用增加虚拟测点，将填补能耗数据转移到新增的虚拟测点下，如图 4.7 所示。



(a)父测点关联变化



(b)所属对象关联变化

图 4.7 拓扑变更时增加虚拟测点

如图 4.7(a)所示，当测点 S 的父测点将发生变化时，新建虚拟测点 V，将测点 S 的基本信息复制给测点 V，然后将测点 S 的父测点置为 F2，最后将测点 S 的所有填补数据转移到测点 V 下。

如图 4.7(b)所示，当测点 P 的所属对象将发生变化时，新建虚拟测点 V，将测点 P 的基本信息复制给测点 V，然后将测点 P 的所属对象置为楼 2，最后将测点 P 的所有填补数据转移到测点 V 下。

4.3 缺失能耗数据填补系统设计

4.3.1 系统设计目标及原则

缺失能耗数据填补系统设计应遵循以下原则：

1）灵活性和扩展性

系统软件的设计应采用模块化结构，使各个模块之间尽量解耦，以达到设置灵活，扩展方便，适应新的发展变化。

2）易用性和易维护性

系统人机界面应简洁友好，操作简单，容易使用。系统的维护应尽量集中、简单，避免复杂的维护开销，减轻维护人员的负担。

3）高效性和节约性

系统的运行响应速度应满足高效性要求，减少系统延迟和用户等待。系统应充分利用存储资源，避免存储资源的过度开销和浪费。

4.3.2 系统整体架构

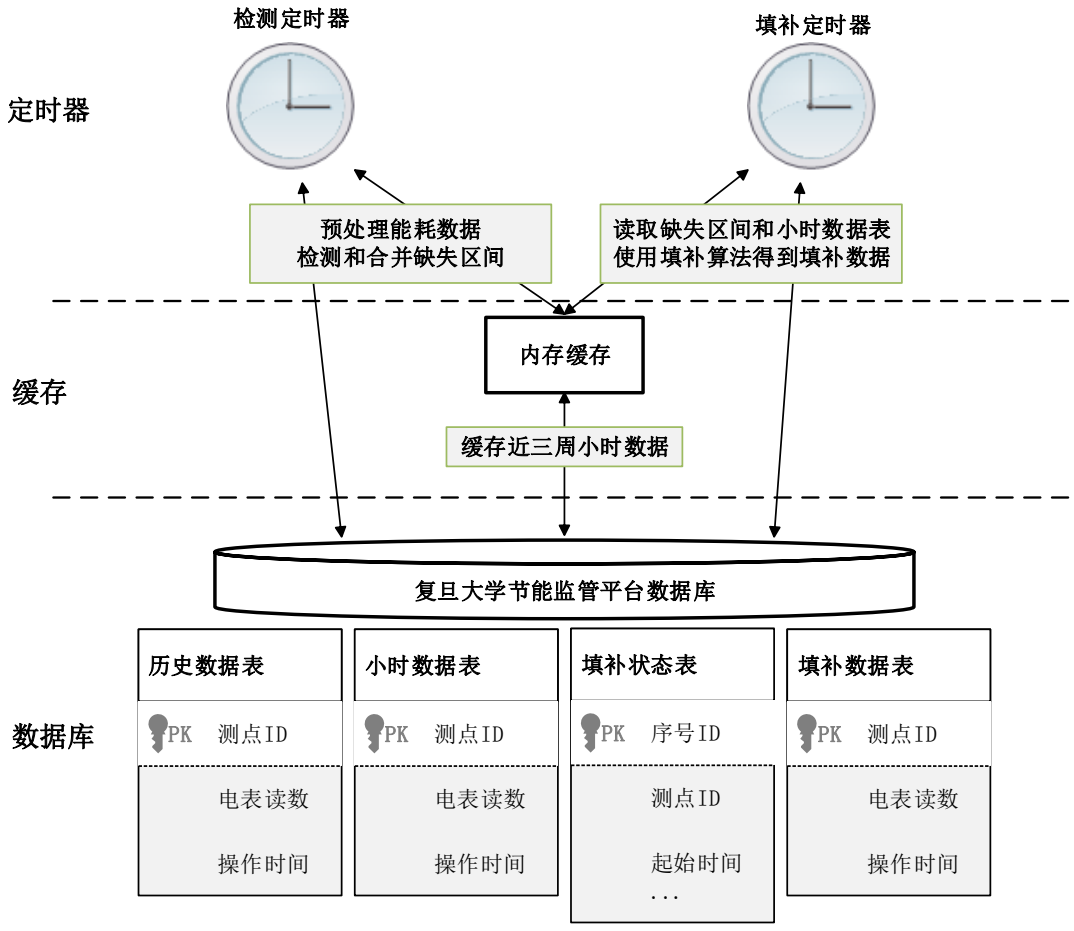


图 4.8 缺失能耗数据填补系统整体架构

如图 4.8 所示，为缺失能耗数据填补的系统整体架构。从系统功能上主要分

为三大部分：定时检测、定时填补和数据预处理，从分层结构上主要分为三层：定时器、缓存和数据库。

4.3.2.1 系统功能模块

为了节省人力成本，系统采用定时机制，自动对缺失数据进行检测和填补。

1）定时检测。复旦大学节能监管平台能耗数据采集的频率为 15 分钟一次，每小时有大约 240 万条记录存入数据库，而提供用户查询的最小粒度为小时。因此，为了防止数据积累过多而导致检测速度过慢，定时检测的频率设为每天 1 次。

2）定时填补。由于用户查询的最小粒度为小时，因此定时填补的粒度为小时。定时填补的频率设为每周 1 次，每次填补上一周的缺失数据。这样做主要有两方面考虑：一是由于网络中断导致缺失的数据，有可能在后来重新上传；二是缺失数据填补需要用到缺失区间前后一周的数据。

3）数据预处理。在检测能耗数据的同时，对能耗数据进行预处理，将分钟粒度的能耗数据处理为小时粒度的能耗数据，并缓存到内存。除了能耗数据，测点间的拓扑结构也同样预先生成，并缓存。

4.3.2.2 系统分层结构

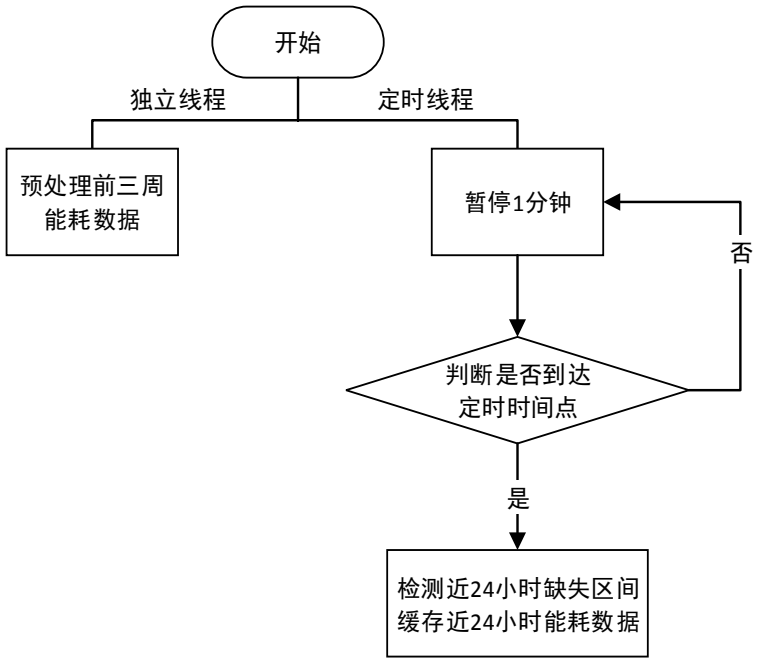


图 4.9 检测定时器处理流程

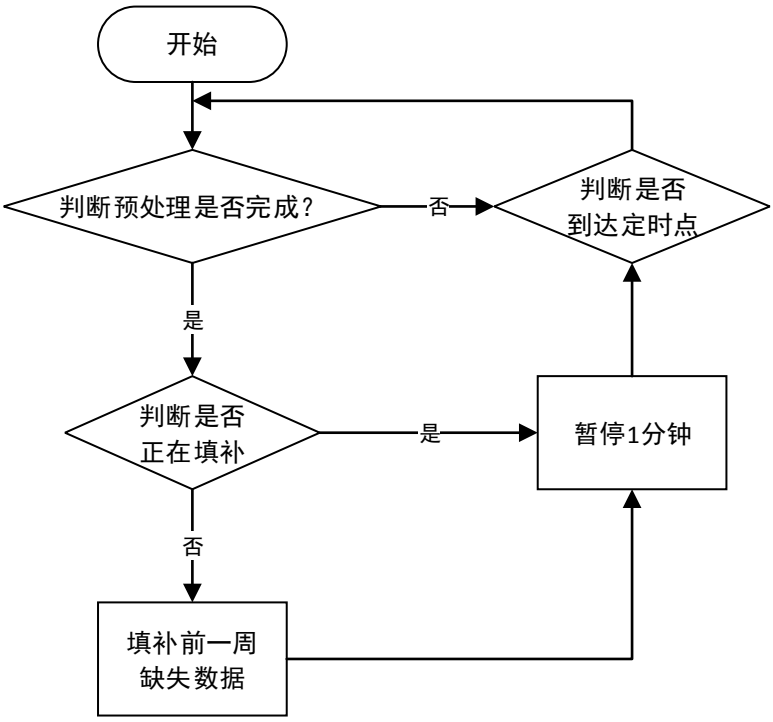


图 4.10 填补定时器处理流程

1）定时器。由检测定时器和填补定时器组成，启动之后的处理流程如图 4.9 和图 4.10 所示。

2）缓存。为了提高填补的效率，将填补时需要的部分数据，缓存到内存。根据算法需要，缓存所有测点近三周的小时能耗数据，以及测点间的拓扑结构等基本信息。

3）数据库。在复旦大学节能监管平台数据库的基础上，新建填补所需的数据表，包括小时数据表（保存测点小时能耗数据）、填补状态表（保存缺失区间和缺失区间的填补状态信息）和填补数据表（保存填补结果）。

4.4 小结

本章首先分析了复旦大学节能监管平台的数据特点，介绍了测点的概念、数据的采集和能耗值的计算方式，对能耗数据的缺失情况进行了分类。然后，根据分析的结果，制定了特定的缺失能耗数据填补策略，针对不同的缺失情况和拓扑的变更，分别制定了填补策略。最后，对填补系统进行了整体架构设计，并对各个模块进行了介绍。

第五章 缺失能耗数据填补工具系统实现

本章基于第四章的系统设计，对缺失能耗数据填补工具进行实现，如图 5.1 所示，为缺失能耗数据填补工具系统实现的技术架构。

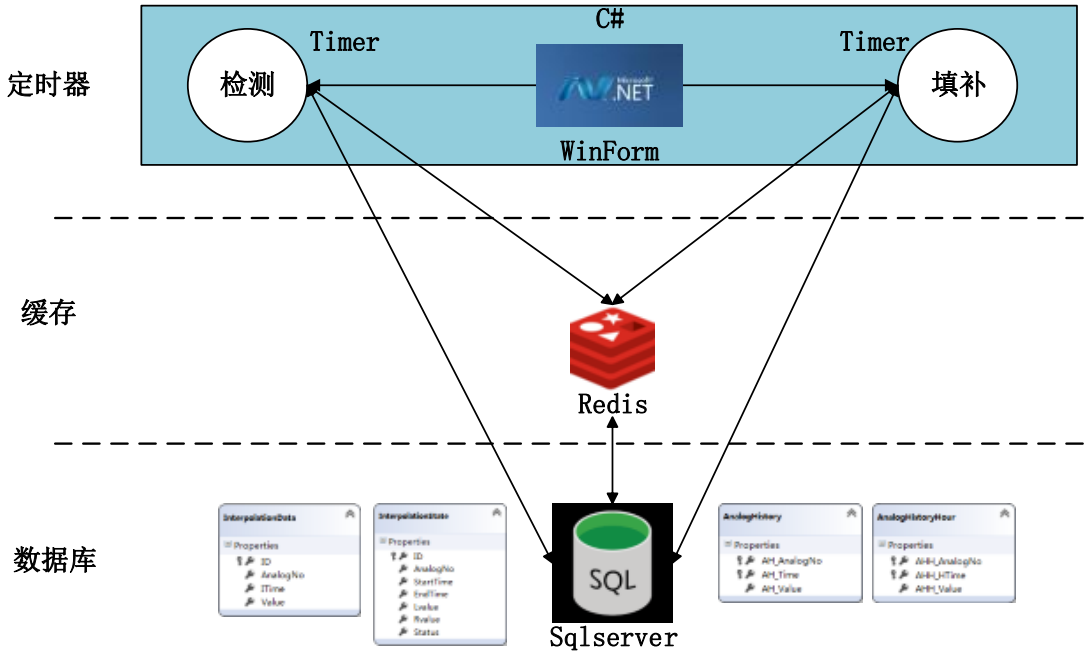


图 5.1 缺失能耗数据填补工具系统实现技术架构

如图所示，使用 C#语言在 Microsoft .Net 开发平台进行开发，以 C/S 作为开发架构，开发 Windows 窗体，采用多线程技术实现定时器，包含检测定时器和填补定时器。在复旦大学节能监管平台数据存储的基础上进行扩展，是用 sqlserver 数据库，新增数据库表用于存储填补信息和数据，使用 linq to sql 对象关系映射组件，对数据库进行操作。在定时器和数据库中间增加缓存层，使用 Redis[33]内存数据库对填补所需的预处理数据进行缓存，以提高填补效率。

5.1 数据存储

复旦大学节能监管平台以 sqlserver 数据库作为存储工具，在此基础上增加数据库表：小时数据表、填补状态表、填补数据表。表定义如下：

表 5.1 小时数据表（AnalogHistoryHour）

字段名	中文名	类型	缺省值	非空	备注
AHH_AnalogNo	模拟量测量点编号	int		Y	主键
AHH_HTime	测量时间	smalldat		Y	主键

		etime			精确到小时
AHH_Value	测量值	float		Y	

表 5.2 填补状态表（InterpolationState）

字段名	中文名	类型	缺省值	非空	备注
ID	序号	int		Y	主键（自增）
AnalogNo	模拟量编号	int		Y	时间序列编号（测点编号）
StartTime	时间	datetime		Y	
EndTime	模拟量值	datetime		Y	
Lvalue	区间左边界读数	float		Y	缺失时，置为负数
Rvalue	区间右边界读数	float		Y	缺失时，置为负数
Status	状态	int			0 表示未填补，1 表示填补完成，-1 表示作废

表 5.3 填补数据表（InterpolationData）

字段名	中文名	类型	缺省值	非空	备注
ID	序号	int		Y	主键（自增）
AnalogNo	模拟量编号	int		Y	时间序列编号（测点编号）
ITime	时间	datetime		Y	粒度为每小时
Value	测量值	float		Y	

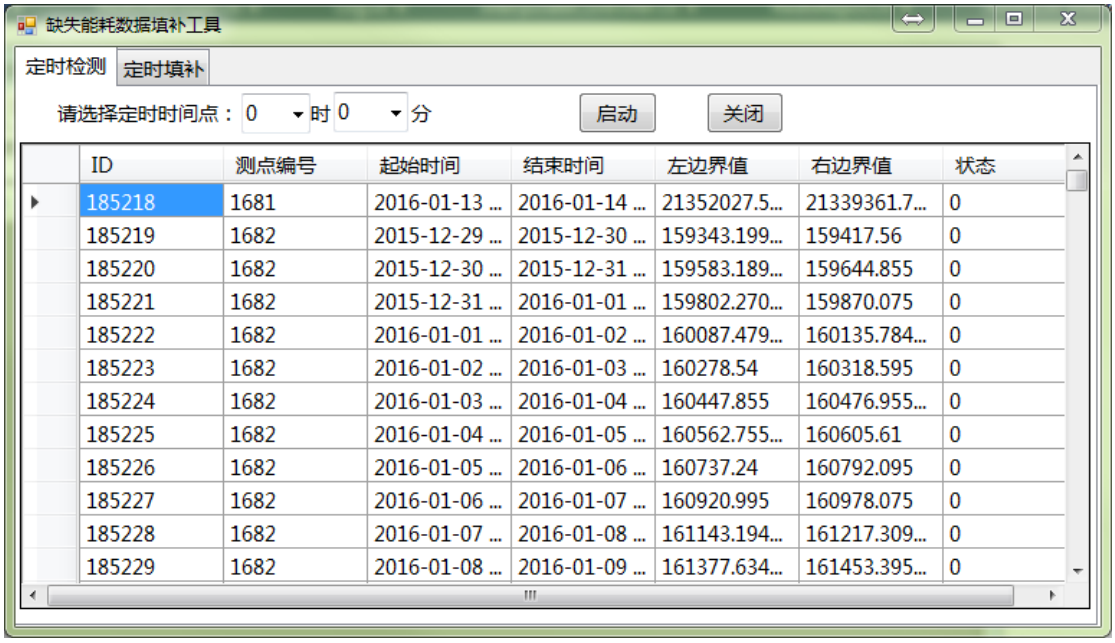
其中，小时数据表用于保存每个测点每小时的能耗值，此处的能耗值为电表读数，因此对于每个测点，能耗值随着时间的递增而递增。填补状态表用于保存每个区间的填补状态，左右边界的读数用于计算区间能耗总和。填补数据表用于保存填补结果数据。

5.2 检测定时器

检测定时器的频率为 1 次/天，并为用户提供选择具体的开始时间点功能。检测定时器主要包含两个主要功能：数据预处理和缺失区间的检测与合并。如图 5.2 所示，为检测定时器的界面，包含小时下拉框、分钟下拉框、启动按钮、关闭按钮和一个缺失区间填补状态表格。

用户可以选择每天检测的定时时间点，点击启动按钮，启动检测定时器。此时，立即开启一个线程，对前三周小时能耗数据进行预处理，并缓存。同时，定时器每分钟对当前时间进行判断，是否到达用户选择的定时时间点，如果是，则检测最近 24 小时缺失区间，并缓存小时能耗数据。点击关闭按钮，关闭定时器。图中的表格，为缺失区间的填补状态，每行对应一个缺失区间，包括序号、测点

编号、起始时间、结束时间、左边界值、有边界值和填补状态等属性。



ID	测点编号	起始时间	结束时间	左边界值	右边界值	状态
185218	1681	2016-01-13 ...	2016-01-14 ...	21352027.5...	21339361.7...	0
185219	1682	2015-12-29 ...	2015-12-30 ...	159343.199...	159417.56	0
185220	1682	2015-12-30 ...	2015-12-31 ...	159583.189...	159644.855	0
185221	1682	2015-12-31 ...	2016-01-01 ...	159802.270...	159870.075	0
185222	1682	2016-01-01 ...	2016-01-02 ...	160087.479...	160135.784...	0
185223	1682	2016-01-02 ...	2016-01-03 ...	160278.54	160318.595	0
185224	1682	2016-01-03 ...	2016-01-04 ...	160447.855	160476.955...	0
185225	1682	2016-01-04 ...	2016-01-05 ...	160562.755...	160605.61	0
185226	1682	2016-01-05 ...	2016-01-06 ...	160737.24	160792.095	0
185227	1682	2016-01-06 ...	2016-01-07 ...	160920.995	160978.075	0
185228	1682	2016-01-07 ...	2016-01-08 ...	161143.194...	161217.309...	0
185229	1682	2016-01-08 ...	2016-01-09 ...	161377.634...	161453.395...	0

图 5.2 检测定时器

5.2.1 数据预处理

数据的预处理主要是将一个时间区间内分钟粒度的能耗数据，转化为小时粒度的能耗数据。由于能耗值是递增的，因此某个小时的能耗值取该小时内最大的分钟能耗数据值，例如 10 点的能耗值，为 9 点-10 点内最大的分钟能耗数据值。转化之后，将缺失的能耗值填充为 0，并将这些能耗数据缓存至内存。最后，提供给缺失区间检测模块扫描缺失区间。主要函数核心代码如代码 5.1：

代码 5.1：数据预处理

输入：

起始时间：*startTime*；

终止时间：*endTime*；

测点编号：*analogNo*；

输出：

小时数据集合：*hourList*

核心代码：

getHourData(DateTime startTime, DateTime endTime, int analogNo)

1. *historyList ← {} , hourList ← {}//historyList 用于保存未填充 0 的小时数据*
2. *historyList ← linq_query(startTime , endTime , analogNo) //使用 linq 语句获取该测点未填充 0 的小时读数*
3. *count ← 0*
4. *//填补缺失时间点读数，将值设为 0.0*
5. *for tmp ← startTime to endTime*
6. *if count >= len(historyList) or historyList[count].time != tmp*
7. *ahh ← {}*
8. *ahh.add(tmp , 0.0)*
9. *hourList.add(ahh)*
10. *else*
11. *hourList.add(historyList[count])*
12. *count++*
13. *end if*
14. *end*
15. *return hourList*

5.2.2 缺失区间的检测与合并

分钟能耗数据经过预处理之后，得到小时能耗数据。然后对小时能耗数据进行缺失区间检测，检测的缺失区间有以下四种情况，如图 5.3 所示，横线表示缺失区间，竖线表示边界没有缺失，即边界能耗值不为 0。

图 5.3 缺失区间情况分类

图 5.3(a)表示当前时间序列两端能耗值不为 0，缺失区间边界没有缺失。图 5.3(b)表示当前时间序列左端能耗值不为 0，右端为 0。图 5.3(c)表示当前时间序列左端能耗值为 0，右端不为 0。图 5.3(d)表示当前时间序列两端都为 0。当且仅当缺失区间边界两端能耗值都不为 0 时，才能得到缺失区间能耗总和。

缺失区间检测完毕后，需要将缺失区间存入数据库，而此时数据库中已经存在缺失区间。因此，当一个缺失区间存入数据库时，与现有的缺失区间存在以下几种关系，如图 5.4 所示，红色线段表示已存在缺失区间，黑色线段表示将要存入数据库的缺失区间。

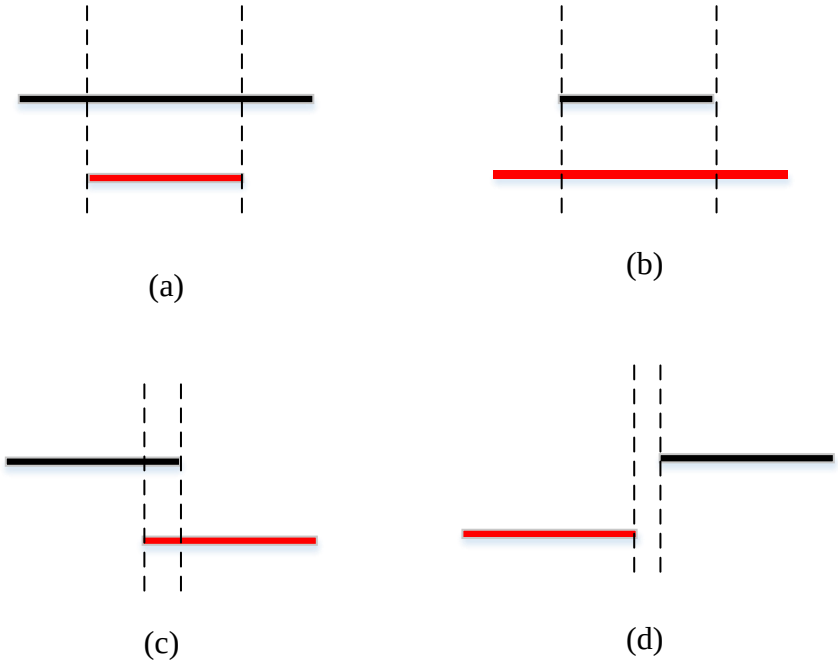


图 5.4 两个缺失区间的关系

图 5.4(a)表示包含关系，将要存入的缺失区间包含已经存在的缺失区间，因此覆盖原有缺失区间。图 5.4(b)表示被包含关系，要存入的缺失区间被已经存在的缺失区间，因此放弃要存入的缺失区间。图 5.4(c)表示相交关系，要存入的缺失区间与已经存在的缺失区间相交，因此将两个缺失区间合并成一个缺失区间。图 5.4(d)表示相离关系，要存入的缺失区间与已经存在的缺失区间相离，因此直接将该缺失区间存入数据库中。

代码 5.2 和代码 5.3 分别为缺失区间的检测与合并的核心代码。

代码 5.2：缺失区间检测

输入：

小时能耗数据：*hourList*

输出：

缺失区间集合：*gapList*

核心代码：

```
scanMissingGap(hourList)
1.  l = -1 , r = -1 , gapList ← {}//保存缺失区间
2.  for i ← 0 to len(hourList)
3.      if hourList[i] == 0 and (i == 0 or hourList[i-1])
4.          l = i
5.      else if hourList[i] == 0 and i > 0 and hourList[i-1] == 0
6.          r = i
7.      end if
8.      if l != -1 and r != -1 or (l != -1 and i == len(hourList)-1)
9.          gapList.add(gap(l , r))
10.         l = -1
11.         r = -1
12.     end if
13. end
14. return gapList
```

代码 5.3：缺失区间合并

输入：

缺失区间集合：*gapList*；

已存在缺失区间集合：*existGapList*；

输出：

合并缺失区间集合：*gapList*

核心代码：

```
mergeGap(gapList , existGapList)
1.  for i ← 0 to len(gapList)
2.      intersectGap ← {} //保存相交的区间集合
3.      for j ← 0 to len(existGapList)
4.          if !(gapList[i].l > existGapList[j].r or gapList[i].r < existGapList[j].l)
5.              intersectGap.add(existGapList)
6.          end if
7.      end
8.      for j ← 0 to len(intersectGap)
9.          gapList[i].l ← min(intersectGap[j].l , gapList[i].l)
10.         gapList[i].r ← max(intersectGap[j].r , gapList[i].r)
11.     end
12. return gapList
```

5.3 填补定时器

填补定时器的频率为 1 次/周，读取缺失区间和小时数据表数据，并对不同的缺失情况和拓扑变更，采用不同的缺失数据填补策略，使用基于约束优化的缺失能耗数据填补算法对缺失数据进行填补。如图 5.5 所示为填补定时器的界面，包含星期下拉框、小时下拉框、分钟下拉框、启动按钮、关闭按钮和一个日志表格。

点击启动按钮，启动填补定时器。此时，立即开启一个线程，判断是否满足填补条件，即预处理数据是否存入缓存。如果满足，则立即对前一周的缺失能耗数据进行填补，否则等到定时时间点进行填补。同时，定时器每分钟判断当前时间是否到达用户选择的定时时间点，如果是，则进行填补。点击关闭按钮，停止定时器。

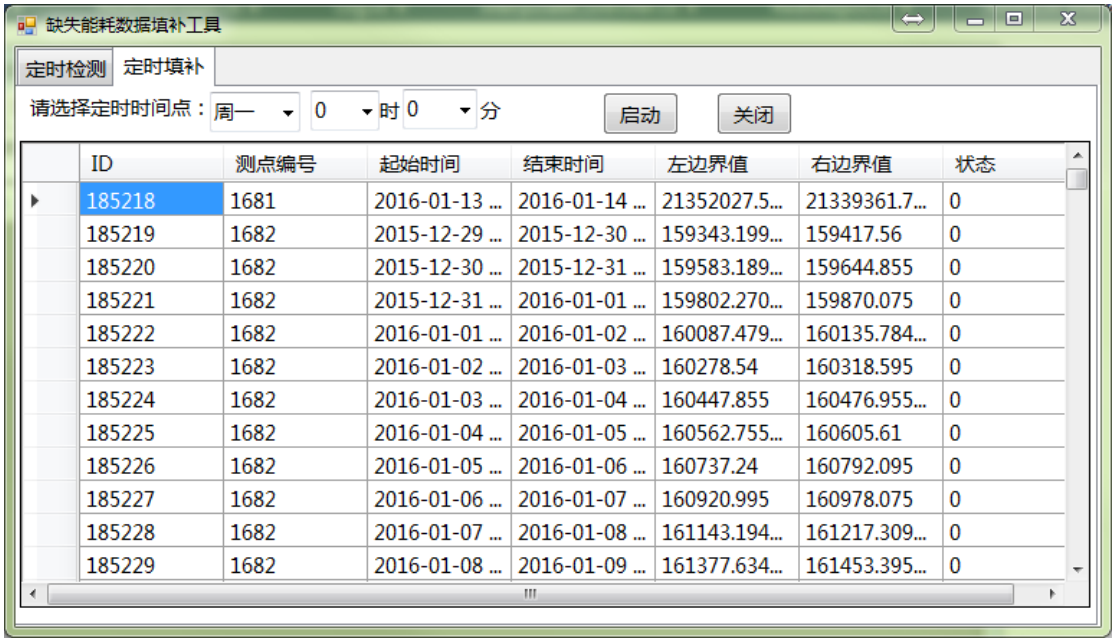


图 5.5 填补定时器

如图 5.5 所示，提供定时时间选择功能，用户可以选择指定每一周的某一天的某个时间点，进行定时填补。同时，为了实现递归填补策略，定义如下数据结构：

```
class PointRelation
{
    private int analogNo;//当前测点 Id
    private int parentNo;//父测点 Id
    private IList<int> sonList;//子测点列表
}
```

5.3.1 递归填补策略实现

实现针对不同缺失情况的递归填补策略，详细过程如代码 5.4 所示。

代码 5.4：缺失数据填补策略

输入：

测点编号：*analogNo*；

缺失区间：*gap*；

测点拓扑关系：*topoRelation={fatherNo:sonList,...}*；

能耗数据缺失信息列表：*missingList={analogNo:gap...}*；

所有测点的数据集合：*seqs*；

输出：

填补结果：*seq*

算法过程：

Strategy(analogNo,gap,topoRelation,missingList,seqs)

1. seq ← {0,0...,0} //用于保存填补结果，初始化为 0

2. sonList ← {}//用于保存子测点列表

3. sonList ← topoRelation.get(analogNo)

4. if sonList.size == 0 //如果没有子测点，那么直接对测点 analogNo 进行填补

5. seq ← Interpolate(analogNo)

6. return seq

7. foreach sonNo in sonList

8. sonGap ← missingList.get(sonNo)

9. if sonGap == null //如果子测点不存在缺失，那么直接取测点真实值

10. sonSeq ← seqs.get(sonNo)

11. else

12. sonSeq ← Strategy(sonNo,sonGap,topoRelation,missingList,seqs)

13. seq ← seq + sonSeq

14. return seq

首先，通过测点的拓扑结构，获取子测点列表，如果没有子测点，则直接对当前测地进行填补。然后，遍历所有子测点，获取子测点缺失区间，如果子测点没有缺失区间，那么将子测点的真实能耗数据，加到当前测点填补结果中，否则对子测点先进行填补（在此处递归），然后将填补值加到当前测点的填补结果中。

最后，将所有子测点的能耗数据之和，作为填补结果返回。

5.3.2 填补条件判断

本文采用定时器对缺失数据进行填补，当每次到达定时时间点时，都会启动一个独立线程进行填补工作。当前必须满足以下两个条件，该线程才能进行缺失数据填补：

1）预处理数据缓存完成。预处理数据是数据填补不可缺失的一部分，因此，在填补线程启动时，首先判断数据的缓存工作是否完成，包括近三周小时能耗数据和测点拓扑结构。

2）当前无其他线程进行填补工作。多个线程同时对缺失区间进行填补，会导致不可预知的错误，造成数据紊乱。因此，在填补之前，必须判断当前是否有线程正在填补。本文采用线程锁 lock[34]，将填补模块代码定义为互斥段(critical section)，来实现多线程对填补代码块的互斥访问，保证任何时候仅有一个线程在执行缺失数据填补任务。C#核心代码如代码 5.5 所示：

代码 5.5：线程互斥核心代码

```
1.  lock (this)
2.  {
3.      IList<InterpolationState> ipsList = getLastWeekGap();
4.      for (int i = 0; i < ipsList.Count; i++)
5.      {
6.          IList<InterpolationData> ipdList = Strategy(ipsList[i]);
7.          if (ipdList != null) interpolationDataRepos.insertList(ipdList);
8.      }
9.  }
```

5.4 缓存

为了提高填补效率，将填补时所需的数据进行预处理，并缓存到内存中。使用 Redis 内存数据库作为缓存工具，Redis 是一个高性能的 key-value 内存数据库

[33]，并提供了各个语言的客户端，使用方便。ServiceStack.Redis 是.Net 驱动类库，支持.Net 对 Redis 的各种操作，需要在项目中添加 4 个引用，ServiceStack.Common 、 ServiceStack.Interfaces 、 ServiceStack.Redis 和 ServiceStack.Text。使用客户端连接池管理 Redis 数据库链接，提升链接速度。

缓存填补所需的预处理数据，包括所有测点近三周的能耗数据和测点的拓扑信息。由于检测定时器的频率是 1 次/天，因此每天更新近三周的能耗数据，如图 5.6 所示，核心代码如代码 5.6 所示。

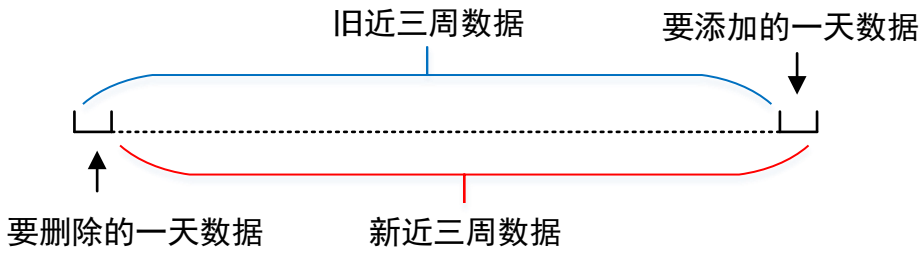


图 5.6 缓存近三周数据

代码 5.6： Redis 操作核心代码

```
1.  ObjectSerializer ser = new ObjectSerializer();
2.  IRedisClient redis = RedisManager.redis;
3.  //1.取出现有数据，从 Redis 数据库中读取数据，并解序列化
4.  IList<AnalogHistoryHour> oldHourList =
    ser.Deserialize(redis.Get<byte[]>(pointList[i].ToString()))
    as List<AnalogHistoryHour>;
5.  //2.更新现有数据
6.  IList<AnalogHistoryHour> newHourList = new List<AnalogHistoryHour>();
7.  for (int j = hourList.Count; j < oldHourList.Count; j++){
8.      newHourList.Add(oldHourList[j]);
9.  }
10. for (int j = 0; j < hourList.Count; j++){
11.     newHourList.Add(hourList[j]);
```

```
12. }
13. //往 Redis 数据库存入数据
14. redis.Set<byte[]>(pointList[i].ToString(), ser.Serialize(newHourList));
```

5.5 小结

本章节详细介绍了缺失能耗填补工具的系统实现，使用 C#语言在.Net 开发平台开发 Windows 窗体程序。定时器层使用 Winform 定时器控件，添加检测定时器和填补定时器，使用多线程技术，实现对缺失能耗区间的检测和填补；数据存储层在复旦大学节能监管平台已有数据库上进行扩展，增加数据库表；缓存层使用 Redis 对预处理数据进行缓存。

第六章 总结与展望

6.1 论文总结

随着能源的日益短缺，我国各地区、各行业的节能意识也在不断增强。由于建筑用能占能耗比例的不断增高，建筑节能成为节约能源中的重要领域。其中，校园建筑能耗具有总量大、种类多等特点，是校园节能中的重要一环。因此，全国逐渐展开节约型校园建筑工作，复旦大学也建立了校园建筑节能监管平台。然而，由于网络异常和设备故障等原因，复旦大学节能监管平台中存在大量缺失能耗数据。对这些缺失能耗数据进行填补，尽可能恢复能耗趋势，能够减少数据盲区，更有利于决策者做出决策。同时维护了能耗数据的完整性，减少对后续能耗分析带来不利影响。因此，本文的主要研究工作包含以下几个部分：

首先，本文介绍了关于缺失数据填补的国内外研究现状，总结了当前已有的缺失数据填补算法存在的不足和缺陷，无法满足区间填补结果的数值总和为固定值的约束。基于国内外研究现状和需求，本文提出了基于约束优化的缺失数据填补算法。

其次，设计了对比实验，使用 MSE、MAE 和 MRE 作为评估指标，将基于约束优化的缺失数据填补算法与均值填补、回归填补和三次样条插值填补进行比较。结果显示该算法有更好的精确度，验证了该算法的精确性和有效性。

然后，充分和详细地分析了复旦大学节能监管平台的数据，总结数据特点，将缺失数据的缺失情况进行分类，并针对不同的缺失情况和拓扑的变更，制定了相应的填补策略。与此同时，对缺失能耗数据填补工具进行了系统设计，将系统分为三层定时器层、缓存层和数据存储层。

最后，对缺失能耗数据填补工具进行了系统实现。使用 C#语言，在.Net 开发平台上开发 Winform 窗体程序，实现了检测定时器和填补定时器。使用 Redis 内存数据库实现缓存层数据存储，对预处理数据进行缓存。数据存储层在复旦大学节能监管平台已有数据存储的基础上进行扩展，新增数据库表，存储填补相关信息和填补结果。

6.2 进一步工作展望

本文的主要研究工作是如何在已知缺失区间和的情况下，对缺失数据进行填

补，并针对复旦大学节能监管平台特定的数据特点，制定填补策略。由于能耗数据的缺失场景复杂多样，而填补算法的选择和填补策略的制定，与缺失场景有着紧密的联系，因此填补算法和策略有值得进一步研究的地方：

1）当遇到缺失区间之间的间隔短的场景。缺失区间的长度有长有短，区间之间的间隔也不固定。基于约束优化的缺失数据填补算法，在计算时间序列相似度时，需要用到缺失区间前一周和后一周的数据。当遇到两个缺失区间之间的间隔很小的场景时，那么用于计算相似度的数据就变得很少，影响相似时间序列的提取，从而影响填补效果。此时，可以考虑两种解决方案：一是改进填补策略，针对该场景使用其他符合一定精度要求，且不需要缺失区间两侧数据作为预处理数据的填补算法来代替。二是对填补算法进行改进，将两个间隔很短的缺失区间作为一个缺失区间进行处理，间隔部分再进行特殊处理。

2）随着数据量的增大，出现填补效率下降的情况。能耗数据的采集是实时不断的，数据量在不断增大。由于填补过程中需要多次检索数据，因此会导致填补效率下降。缺失数据中有很多短期缺失数据，即缺失区间小于 24 小时的缺失数据，甚至有 2-3 小时以内的及短期的缺失区间。如果此时填补速度很慢，那么填补的性价比就变得很低。在这个情况下，可以改进填补算法的实现过程，减少对数据的检索。另外可以考虑从并行计算方面进行效率上的改善。

参考文献

[1] 郑明明, 陈硕. 建筑能耗监测平台的研究[J]. 建筑节能, 2009, 37(9):53-55.

[2] 马金星. 节约型校园节能监管平台关键技术开发与建筑能耗特性评价[D]. 大连理工大学, 2011.

[3] 杨石, 罗淑湘, 钟衍,等. 大型公共建筑能耗监测平台存在问题及其初步解决方案[J]. 建筑技术, 2014, 45(8):714-718.

[4] Pan L, Li J. K-Nearest Neighbor Based Missing Data Estimation Algorithm in Wireless Sensor Networks[J]. Wireless Sensor Network, 2010, 2(2):115-122.

[5] Bennis S, Berrada F, Kang N. Improving single-variable and multivariable techniques for estimating missing hydrological data[J]. Journal of Hydrology, 1997, 191(1-4):87-105.

[6] Tang W Y, Kassim A H M, Abubakar S H. Comparative studies of various missing data treatment methods - Malaysian experience[J]. Atmospheric Research, 1996, 42(1):247-262.

[7] Linear Interpolation[J]. 2006:49-69.

[8] Wang F M, Li-Xia H U. A Missing Data Imputation Method Based on Neighbor Rules[J]. Computer Engineering, 2012, 38(21):53-55.

[9] Berrut J P, Trefethen L N. Barycentric Lagrange interpolation[J]. Siam Review, 2004, 46(3):501-517.

[10] Wolfram MathWorld . Lagrange Interpolating Polynomial[EB/OL] , http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html , 2017.

[11] Baltazar-Cervantes J C. Study of Cubic Splines and Fourier Series as Interpolation Techniques for Filling in Short Period of Missing Building Energy Use and Weather Data[J]. Journal of Solar Energy Engineering, 2006,128(2):15-21.

[12] Chen H, Claridge D E. Procedures for Filling Short Gaps in Energy Use and Weather Data[J]. [660]Symposium on Improving Building Systems in Hot and Humid Climates, 2000.

[13] Hu J, Ogunsola O T, Song L, et al. Restoration of 1–24 hour dry-bulb temperature gaps for use in building performance monitoring and analysis—Part I[J]. Hvac & R Research, 2014, 20(6):594-605.

[14] Yi X, Zheng Y, Zhang J, et al. ST-MVL: Filling Missing Values in Geo-sensory Time Series Data[C]. 2016.

[15] Lecun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015, 521(7553):436.

[16] Schmidhuber J. Deep Learning in neural networks: An overview.[J]. Neural Networks the Official Journal of the International Neural Network Society, 2015, 61:85.

[17] Sozykin A V. An overview of methods for deep learning in neural networks[J]. Ural Federal University (Mira str. 19, 2017:28–59.

[18] Coulibaly P, Evora N D. Comparison of neural network methods for infilling missing daily weather records[J]. Journal of Hydrology, 2007, 341(1):27-41.

[19] Xue-Yong Y E, Jun-Ji W U, Yang W, et al. The Correction of the Bad Data in Power System Based on Neural Network[J]. Power System Technology, 2007, 52(5):692-696.

[20] Yozgatligil C, Aslan S, Iyigun C, et al. Comparison of missing value imputation methods in time series: the case of Turkish meteorological data[J]. Theoretical & Applied Climatology, 2013, 112(1-2):143-167.

[21] Se-Hark Park. Decomposition of industrial energy consumption : An alternative method[J]. Energy Economics, 1994, 17(4):265-270.

[22] 欧育辉, 刘轶芳, 满讲义. 基于 LMDI 的我国能耗增长总量分解[J]. 经济管理, 2007(7):91-95.

[23] Niu Q F, Zhang Y J, Zhang C H. Energy Split Method in Building[J]. Control Engineering of China, 2010.

[24] 王莎. 能耗拆分算法在大型公建分项监测系统中的应用研究[D]. 湖北工业大学, 2011.

[25] 闭小梅, 闭瑞华. KNN 算法综述[J]. 科技创新导报, 2009(14):31-31.

[26] Yabea . K-近邻（KNN）算法[EB/OL] , https://www.cnblogs.com/ybjourney/p/4702562.html , 2015.

[27] 阿凡卢 . Dynamic Time Warping 动态时间规整算法[EB/OL] , https://www.cnblogs.com/luxiaoxun/archive/2013/05/09/3069036.html , 2013.

[28] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In Proc. ACM KDD Workshop,pages 359–370, 1994.

[29] 恩格尔布雷希特. 计算群体智能基础[M]. 清华大学出版社, 2009.

[30] 龙信数据 . 缺失数据的插补方法简述[EB/OL] , http://blog.sina.com.cn/s/blog_66239fdb0101kfqw.html , 2013.

[31] 岳勇, 田考聪. 数据缺失及其填补方法综述[J]. 预防医学情报杂志, 2005, 21(6):683-685.

[32] 西点的学生 . 回归法插补缺失值[EB/OL] , https://wenku.baidu.com/view/bd797567aaea998fcc220eef.html , 2014.

[33] 黄健宏. Redis 设计与实现[M]. 机械工业出版社, 2014.

[34] Microsoft Developer Network . lock 语句[EB/OL] , https://msdn.microsoft.com/zh-cn/library/c5kehkcz(VS.80).aspx , 2017.

致谢

研究生生涯一晃而过，回首过去，一路走来收获颇多。还记得三年前保研面试之后，幸运地被顾宁老师招进实验室。第二年 3 月份就到实验室写本科毕业论文，和我一起到实验室做毕业设计的还有丁海洋同学和曹浩哲同学，在那时我们就结下了深厚的友谊。由于比其他同学早来实验室，因此让我对实验室有更强烈的自豪感和亲切感。直到暑假，我们这一届的同学都到齐了，拖着行李箱，一起住在了学校附近的宾馆。每天早上顶着炎热的太阳，往返于宾馆和学校之间，一起吃饭，一起去本部办理校园卡。所有的一切都历历在目，永远深深地留在我的记忆深处。

首先，我要感谢我的导师丁向华副教授，丁向华老师认真细致的治学态度和求实执着的科研精神，是我今后工作和学习的榜样。

其次，我要感谢顾宁老师对我的指导与教诲，一到实验室，顾宁老师就给我们灌输求真务实的科研精神，顾宁老师严谨的工作科研态度深深地影响了我。

然后，我要感谢卢暾老师对我的帮助和鼓励，在确定毕业论文选题和完成论文期间，卢暾老师都给了我很多意见和建议，对我的论文写作非常有帮助。

最后，我要感谢实验室的同学们，感谢你们在我困难时给予我的帮助与鼓励。感谢刘鹏师兄在论文开题时给我的帮助。特别感谢张鹏师兄在我论文选题和写作过程中给予我的帮助，在完成论文之前，我曾经多次找张鹏师兄咨询和讨论相关问题，张鹏师兄都给了我很好的建议和意见，受益良多。感谢我的室友三年来对我的帮助和包容。

我想用最朴素的语言，表达我最真挚的感谢，谢谢你们！

论文独创性声明

本论文是我个人在导师指导下进行的研究工作及取得的研究成果。论文中除了特别加以标注和致谢的地方外，不包含其他人或其它机构已经发表或撰写过的研究成果。其他同志对本研究的启发和所做的贡献均已在论文中作了明确的声明并表示了谢意。

作者签名：[signature] 日期：2018.6.5

论文使用授权声明

本人完全了解复旦大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存论文。保密的论文在解密后遵守此规定。

作者签名：[signature] 导师签名：[signature] 日期：2018.6.5