Vol.2020-ITS-80 No.10
2020/3/10

バス時刻表改訂のためのデータモデルとアーキテクチャの提案

舞田 涼太朗[2,a)]　山口　翼[2]　川谷　卓哉[2]　峯　恒憲[3,b)]

概要：本稿は，情報処理学会論文誌ジャーナルに投稿する原稿を執筆する際，および論文採択後に最終原稿を準備する際の注意点等をまとめたものである．大きく分けると，論文投稿の流れと，LaTeX と専用のスタイルファイルを用いた場合の論文フォーマットに関する指針，および論文の内容に関してするべきこと，するべきでないことをまとめたべからずチェックリストからなる．本稿自体も LaTeX と専用のスタイルファイルを用いて執筆されているため，論文執筆の際に参考になれば幸いである．

キーワード：バスプローブデータ，時刻表改訂，アーキテクチャ，データモデル

Data Model and Architecture for using Bus Probe Data toward Revising Bus Time Tables

Ryotaro Maita[2,a)]　Tsubasa Yamaguchi[2]　Takuya Kawatani[2]　Mine Tsunenori[3,b)]

Abstract: Toward revising bus time tables, we have collected bus probe data since 2019 and analyzed them to estimate bus travel time or delay time. In this paper, we discuss data model and architecture for using bus probe data so as to revise bus time tables.

Keywords: Bus Probe Data, Time Tables Revision, Architecture, Data Model

1. はじめに

近年カープローブデータの活用事例が増えてきている. カープローブデータとは少なくとも日時と位置を含み, プローブ機器によって速度, 加速度, ドライブレコーダなどのカメラ画像などの情報をいくつか含む.

バス運行の分野では, 遅延時間分析や利用者状況の分析などに応用され, ダイヤグラム改正などに利用されている [1] しかし, バス運行におけるプローブデータをバスの時刻表や便番号, 経路と連動してアクセスする機構は統一的に定まっておらず, 各社の持つプローブデータやバス会社のもつ時刻表などのバス情報を各々照会して結合する実装を各社が行う必要がある.

本研究では, 各社の持つバス情報の抽象構造とプローブデータと連動してアクセスするためのロジックとデータモデルを構築することを目的とし, 異なる形式のバス情報を持つ情報提供者及び, 異なる利用目的のあるデータ利用者にとって統一的に扱えるモデルを考察した. ケーススタディとして, 昭和自動車の路線バスを対象として時刻表改訂を目的とした適用を行った.

2. 関連研究

公共バスのプローブデータを利用した公共バスの利便性の向上への取り組みとして, ダイアグラムの改定 [1] や, 一

[1] 情報処理学会
IPSJ, Chiyoda, Tokyo 101–0062, Japan
[2] 九州大学大学院システム情報科学府
[3] 九州大学大学院システム情報科学研究院
[†1] 現在，情報処理大学
Presently with Johoshori University
[a)] maita.ryotaro@m.ait.kyushu-u.ac.jp
[b)] mine@ait.kyushu-u.ac.jp

般利用者へ提供する予測所要時間の算出 [2] などが行われている. 各取り組みにおいて, プローブデータの収集機器の違いや, バス会社の提供する時刻表データ形式の違いを吸収するための前処理を行っている.

バスの時刻表などの静的データや車両の位置などのリアルタイムな情報を規格化し, 公共バスデータのオープン化及びデータを利用する際の業務上の手間を減らす目的として, 国土交通省は公共交通機関の時刻表と地理情報に関するオープンフォーマットである General Transit Feed Specification(GTFS)[3] をもとにした, 標準的なバス情報フォーマット [4] を定めている. しかし, 車両のリアルタイムな情報の規格については, 便情報, 位置情報, バス停の通過情報など一般利用者にとって必要な最低限の情報にとどまり, バス車両に取り付けた機器から収集されたプローブデータを統一的に扱う規格にはならない.

プローブデータのような時間, 座標に紐づく種々のデータに対し統一的かつ高速なアクセスを実現する取り組みとして, 地理情報時空間データベース [5] が提案されている. これにより, プローブデータの持つ交通情報, 車両情報の種類によらず, 時刻や座標に関する色々な検索条件を利用してデータを検索することが可能であるが, 公共バスの特定の運行区間上のデータの抽出など特定のデータの特性に基づく条件抽出を行うには, 汎用性を落として特定のデータを扱えるようなデータベース設計を行うか, 特定のデータ特性を利用した条件抽出を別レイヤで実装する必要がある.

本研究では種々のプローブデータについて時刻表情報との対応付けを行い, 統一的にデータ抽出をできるようなデータモデル及びアーキテクチャの考案を目指したが, 類似の研究として, 時系列データを, 時系列クラスタに転換するとともに，そのクラスタから時間依存の連想規則を抽出するフレームワークを提案, 実装した研究がある [6]. この研究では未加工の衛星画像時系列データについて, 季節や気象の前後関係を表すクラスター及びクラスターと元の時系列データの区間との対応をデータベース化することで,「梅雨の期間の衛星画像」「台風が去った後の 3 か月以内の衛星画像」などの条件抽出がデータベースに対するクエリによって表現できることを示した. また, この事象関係のクラスタリング, クラスタと時系列データの対応のデータベース化, データベースへのクエリによる条件抽出というフレームワークが, 音声, 映像といった異なる領域の時系列データにも適応可能であることを考察している.

3. 本研究の意義

本研究では, 公共バスの各プローブデータの形式の違いや格納しているデータの違い, または計測頻度の差を吸収して, プローブデータとバス停の運行情報と時刻表との対応情報を保存し, 時刻表や運行情報を検索条件に利用した各プローブデータへのアクセスを実現するアーキテクチャを提案する. これには,「形式や内容の異なるプローブデータ, もしくは異なる頻度で計測されたプローブデータ同士を, 同一のアーキテクチャで扱うことを可能にする」という意義がある.

通常, バスプローブデータを取得する機器から, 直接バス運行の時刻表と対応付けがなされたデータを取得できることはできない.. よって, プローブデータ計測機器によって計測し, 集計したデータについて各バス会社の時刻表や運行データと照らし合わせる処理を行うのが一般的である.

しかしその実装は, 公共バス車両に取り付けられた機器から会社専用の運行情報センターと情報のやり取りをして自動化する場合や, 運転手が日報に人力で記載した車両 ID と運行系統の情報を利用するもの, もしくはそのような対応付けをプローブデータと時刻表のデータから直接総当り的に行うなど, 各事情に応じた対応がなされており, 汎用性を考慮した規格化がなされてこなかった.

そこで本研究では, 各プローブデータの形式を抽象化し, また各公共バス会社によって一般的に用意されると想定される「時刻表データ」,「仕業データ」,「バス停座標データ」のみを利用して, プローブデータと時刻表データの関係を保存しアクセスする汎用的に利用できるデータモデルとアーキテクチャを提案する.

4. 提案するアーキテクチャの概要

本アーキテクチャの概要を図 1 に示す. 要点は以下の二つである.

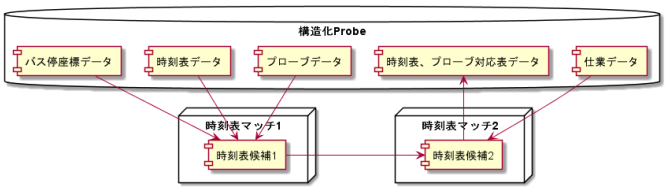

図 1 アーキテクチャ構成

(1) プローブデータ, 時刻表や仕業表などの公共バスの運行情報データ, およびプローブデータと運行情報対応表データの仕様設計 (これらを合わせて本研究では構造化 Probe と呼ぶ)
(2) プローブデータ, 時刻表や仕業表などの公共バスの運行情報データのみから, プローブデータと運行情報対応表データの作成するロジックの提案

次章から,(1),(2) の詳細について述べる. また, 各データを保存する形式は, 地理情報データベースとして使われているオープンソースのリレーショナルデータベースシステム (RDBMS) である PostgreSQL および PostGIS[7] に準拠した.

5. 構造化 Probe の仕様設計

本アーキテクチャでは, 各プローブデータの仕様やバス会社の持つ時刻表や仕業表のデータの違いを吸収し, 汎用化するため, 構造化 Probe 内の仕様には, 特殊な場合でしか利用できないようなデータを設計に含めないように留意した.

5.1 バスプローブデータのデータ形式

バスプローブデータのデータ形式を表 1 に示す. 公共バスのバスプローブデータ計測機器は, 車両と 1 対 1 に対応していると考えられる. また, プローブデータの持つ時刻や座標のデータは設計に含めても汎用性を失うことはないが, それ以外にどのような情報を含むのかは目的や観測機器によって変わるので, 任意の形式のデータの追加が可能となるよう設計した.

5.2 時刻表のデータ形式

時刻表のデータ形式を表 2 に示す. 時刻表のデータ形式は, バス会社から一般に提供される時刻表に含まれると想定されるデータのみを設計に含めた. また時刻表の改訂が行われた場合でも, あらたなデータベースの設計を行う必要が無いように, 改訂された時刻表情報は蓄積される設計にした.

5.3 バス停座標のデータ形式

バス停座標のデータ形式を表 3 に示す. バス停の座標は時刻表の改定に伴い追加, 削除, 移動される場合があるので, 時刻表のに伴いバス停座標のデータが蓄積される設計にした.

5.4 仕業表データの形式

仕業表データの形式を表 4 に示す. 公共バスは, 一台の車両がどの便をどの順序で運航するかが決められていることが一般的である. これらは仕業データとよばれ, 各運行便の順序に各仕業名がついている.

5.5 時刻表とプローブデータの対応を表すデータ形式

時刻表とプローブデータの対応を表すデータ形式を表 5 に示す. このデータは,5.1,5.2,5.3,5.4 で述べたデータをもとに、プローブデータに対して時刻表上のどの便のバス停区間の物であるかの対応関係を保存するものである. 詳細なアルゴリズムは 6 にて述べる.

6. プローブデータと運行情報対応表データの作成するアルゴリズム詳細

アクセスを提供するデータベースを構築する手順として,

(1) プローブデータと時刻表から便候補を出す

表 1 バスプローブデータのデータ形式

項目	データ型
車両 ID	integer
走行日時	integer
座標	ST_Point
任意のデータ	任意

表 2 時刻表のデータ形式

キー属性	項目	データ型
主キー	改定日時	integer
	便番号	integer
	停車バス停番号	integer
非キー	停車時刻	integer
	バス停名	text

表 3 バス停座標のデータ形式

キー属性	項目	データ型
主キー	改定日時	integer
	バス停名	text
非キー	バス停座標	ST_Point

表 4 仕業表のデータ形式

キー属性	項目	データ型
主キー	改定日時	integer
	仕業名	text
非キー	便番号リスト	[integer]

表 5 時刻表とプローブデータの対応を表すデータ形式

キー属性	項目	データ型
主キー	運航日	date
	便番号	Integer
	停車バス停番号	Integer
非キー	車両 ID	Integer
	通過時刻	TimeStamp

(2) 仕業データを利用して便候補から仕業を特定するの 2 つを行う.

6.1 プローブデータと時刻表から便候補を計算する

5 で述べた,「バスプローブデータ」,「バス停座標のデータ」を利用して, プローブデータの全点に対し一定距離以下の最も近いバス停を求める. この際の距離の決め方は, プローブデータの計測間隔により調整する必要がある. 次に「時刻表データ」を利用して, プローブデータの全点に対し, 最も近いバス停と時刻表内の便のバス停と比べて,n 秒以内のずれにある便を, 「候補運航便→プローブ点シーケンス」の辞書に追加し, プローブデータ列が候補となった運航便とどれだけマッチしているかのマッチ度を計算する. マッチ度の計算方法は,「プローブ点シーケンスに含まれるバス停名列と候補運航便の通るバス停列名の最長共通部分列の要素数/候補運航便の通るバス停列名」で求める. マッチ度を計算したのち, 候補運航便をマッチ度の高い順にソートし, 運航便の時刻区間の重複がないように選択する。疑似

コードを Algorithm 1 に示す.

```
Input:
    var Probes : [{Time: time,
        NearestBusStopName :str},
        NearestBusStopDistance : float]
    var Diagram : Map(ServiceId : int,{
        BusStopNames : [str] , StopTimes[
        time]})
    var MaxGapTime : int
    var MaxDist : float
Output:
    CandidateScores : [{ServiceId : int,
        Score : float,StartTime: time,
        EndTime: time}]
Definition:
    var candidates : Map(ServiceId :int,{
        BusStopName : str,Time :time})
    for probe : {time,str,float} in Probes:
        if probe.NearestBusStopDistance >
            MaxDist
            continue
        for service : {int,[str],[time]} in
            Diagram:
            for stoptime : time in service.
                StopTImes
                if abs(probe.Time - stoptime)
                    < MaxGapTime
                    candidates[serviceId].
                        append(probe.
                        NearestBusStopName,
                        probe.Time)
    var candidateScores : [{ServiceId : int,
        Score : float,StartTime: time,
        EndTime: time}]
    for serviceId : int ,busStopSequence :
        [{BusStopName : str, Time : time}]
        in candidates
        var score : float = LCS(Diagram[
            serviceId],BusStopNames).length /
             Diagram[serviceId].BusStopNames.
            length
        candidateScores.append(serviceId,
        score,
        busStopSequence.first.Time,
        busStopSequence.last.Time)
    return candidateScores
```

Algorithm 1

6.2 仕業データを利用して便候補を絞り込む

一般にバス車両がどの仕業を運行するかが常に決まっているわけではない. なぜなら, 車両と仕業の対応付けを行ってしまうと, 遅延等の理由で次の仕業が定刻に運行できない場合などで例外が発生してしまうためである. よって, バス車両と仕業は運航日ごとに求める必要がある. 「仕業表データ」を利用することで,6.1 によって計算された運航便の属する仕業を求め, ある同一の仕業に含まれる便が一定以上含まれている場合, 車両はその仕業を運行したものとする. この処理により, 該当する仕業は複数になる可能性もある.

7. 昭和バスプローブアーキテクチャ構成

本研究では, バス情報及びプローブデータを格納するリレーショナルデータベースマネジメントシステム (RDBMS) として PostgreSQL および PostGIS を採用した. 全体図を図 2 に示す

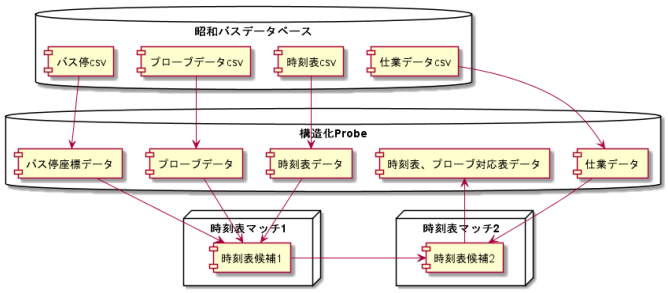

図 2 アーキテクチャ構成

7.1 マッチアルゴリズムの評価

6.1 および 6.2 で示したマッチアルゴリズムを, 昭和バスの 10 月中の運行プローブデータと時刻表データに適用した.

最初に, マッチアルゴリズム 1 による仕業推定の結果を表 6 に示す. ただし, 使用したプローブデータは著しい計測不備のあったものを除いた. 表 6 より, マッチアルゴリズム 1 によって, プローブデータから運行仕業を完全に特定できたのは全体の仕業数の 11%であった. また, 各日付の各仕業において, 正答すべき便数のうち, 候補として挙げられたものの中に含まれていた割合は, 平均 89%であり, 各仕業において候補洩れはが平均 11%存在した. また, 候補便のうち実際には運行していない不正解便が平均 18%含まれていた. 不正解便が発生するケースは,

(1) 実際には回送便だったが, 同時刻に似た経路をたどる便が時刻表に存在した
(2) 遅延のため, 正解となる便と運行時刻が近い便と混同した

に分類できた

次に, アルゴリズム 2 による仕業推定の結果を表 7 に示す.

アルゴリズム 2 の性質上, 正解の仕業データを利用してプローブデータと時刻表の対応を行うので, 対応候補の中

表 6　アルゴリズム 1 による候補精度

運行した総仕業数	547
完全マッチした仕業数	61
完全マッチした仕業数割合	0.11
候補に含まれる平均正解便割合	0.81
候補に含まれる平均不正解便割合	0.18
正解すべき便のうち, 候補に挙げれた便数の割合 (recall)	0.89

表 7　アルゴリズム 2 による候補精度

運行した総仕業数	547
完全マッチした仕業数	535
完全マッチした仕業数割合	0.99
候補に含まれる平均正解便割合	0.99
候補に含まれる平均不正解便割合	0.00
正解すべき便のうち, 候補に挙げれた便数の割合 (recall)	0.99

には正解仕業以外の便は現れず,「候補に含まれる平均不正解便の割合」は常に 0 になることに注意する. アルゴリズム 2 では, プローブデータから運行した仕業の 99%で正しく仕業を構築でき, 各仕業において運航便の候補洩れは平均 1%未満となった. ケーススタディでは昭和バスのデータを利用したが, アルゴリズム 1, アルゴリズム 2 で使用した時刻表のデータや仕業のデータは昭和バスに特化したものでなく汎用的な形式のものであり, またプローブデータにおいても一般に想定される地理情報, 時刻情報, 車両情報のみを用いたので, 実際のバスプローブデータに対し時刻表との対応付けを十分な精度で行うことができることが示された.

7.2 時刻表と対応したバスプローブデータへのアクセスの評価

本アーキテクチャによって時刻表とと対応した区間のプローブデータへのアクセスが実現されていることを示す.Algorithm 2 に, 本アーキテクチャ上で,「経路によらず二点のバス停間を通るプローブデータ」を求める際の疑似コードを示す.

```
1    with candidateroutes as (SELECT d1.
         runnumber,d1.stopnumber as start,d2.
         stopnumber as end FROM diagram as d1
2  INNER JOIN diagram as d2
3          ON d1.runnumber = d2.runnumber
4  where d1.stopname = $1 and d2.stopname = $2
       and d1.stopnumber < d2.stopnumber)
5  , routelist as (select s1.date,s1.busname,s1
       .runnumber,s1.runtime as starttime, s2.
       runtime as endtime from stoptime as s1 ,
       stoptime as s2, candidateroutes where
6  s1.date = s2.date and
7   s1.busname = s2.busname and
8   s1.runnumber = s2.runnumber and
9   s1.runnumber = candidateroutes.runnumber and
10  s1.stopnumber = candidateroutes.start and
11   s2.stopnumber = candidateroutes.end)
12 select * from routelist,probe where
13 probe.busname = routelist.busname and
14 probe.date = routelist.date and
15 probe.runtime between routelist.starttime and
       routelist.endtime;
```

Algorithm 2

またこの処理の後に, 具体的なバス車両や便の経路, 仕業を特定することでより詳細な抽出を行うことができる.

8. まとめ

本研究によって, プローブデータの形式や各バス会社の時刻表データや運行データの形式の違いを吸収し, プローブデータと時刻表上との対応づけを行うことができた. 今後は、実際にこのアーキテクチャを利用して, 時刻表改定へのデータ分析, 抽出を行うことを目指す.

謝辞　本研究で利用したデータは，昭和自動車より提供を受けた．また，本研究の一部は，科研費（課題番号 JP15H05708）の支援を受けた．ここに感謝いたします．

参考文献

[1] Tohei, O., Keita, M., Kiyoshi, H., Yusuke, M. and Masaki, I.: IMPROVING BUS SCHEDULES BASED ON REAL-TIME BUS ARRIVAL INFORMATION WITH OPEN INNOVATION IN RYOBI GROUP, 第 57 回土木計画学研究発表会 (2018).

[2] 浩　辰巳，雄作大野：バスプローブデータを用いた路線バスの予想所要時間に関する基礎的研究，都市政策研究，No. 9, pp. 79–86（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/40017084287/⟩ (2010).

[3] : GTFS, https://developers.google.com/transit?hl=ja.

[4] : 標準的なバス情報フォーマット　ガイドライン，http://www.mlit.go.jp/sogoseisaku/transport/sosei_transport_tk_000112.html/.

[5] 蔵之花舘，達郎木村，宣宏　沖，直子重松，磯生上野，一兵衛内藤，貴司久保，和大宮原，淳　磯村：高速時空間データ管理技術「Axispot」と時空間データ高速検索技術 (特集 実世界の事象をデータ化しながら活用するフィジタルデータセントリックコンピューティング)，NTT 技術ジャーナル，Vol. 31, No. 11, pp. 18–22（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/40022071744/⟩ (2019).

[6] Honda, R. and Konishi, O.: Temporal Rule Discovery for Time-Series Satellite Images and Integration with RDB, *Principles of Data Mining and Knowledge Discovery* (De Raedt, L. and Siebes, A., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 204–215 (2001).

[7] : PostGIS, http://postgis.refractions.net.