M-002

GTFS データを用いた路線バスの遅延状況推定
Delay Estimation for Route Bus on GTFS Data

但馬　慶行†　　高橋由泰†
Yoshiyuki Tajima　　Yoshiyasu Takahashi

1. はじめに

次世代交通システムとして Mobility as a Service(MaaS)が注目されている[1]．MaaS はバス，鉄道，タクシーなどの交通機関を情報技術で結び付け一つのシームレスなサービスとしてエンドユーザに提供しようとするものである．例えば，旅行のためにある目的地へ行きたい場合，スマートフォンから目的地を検索すると鉄道だけでなくバスやタクシーなどを含めた旅程の候補が表示され，ひとたびある旅程を選択したならば，予約，決済がなされ，案内に従って各交通機関を利用できる，というものである．このようなサービスは，実行可能性を考慮しなければ比較的に実現は容易である．しかしながら，実行可能な旅程をユーザに提案するためには，遅延やリソースなどを鑑みる必要がある．特に路線バスは一般の道路網を利用しているため，時々刻々と変化する遅延を的確に把握することが不可欠となる．

一方，国内の路線バスに関して，すでにバスロケーションシステムの導入が進みつつある．さらに，General Transit Feed Specification (GTFS) [2]や GTFS-realtime(GTFS-RT) [3]を拡張した標準的なバス情報フォーマット(GTFS-JP) [4]に則って，データが公開されはじめている．例えば，路線図，時刻表などの静的なデータとバスの位置などの動的データが入手できるようになっている．このような広く知られたデータフォーマットに基づいて，技術開発やサービス構築を進めることは，導入コストを引き下げ普及を加速するために重要である．

そこで本研究では，路線バスを対象として，GTFS および GTFS-RT のデータ(GTFS データと呼ぶ)を使って時々刻々と変化する遅延状況を推定する方法を提案する．

2. 関連研究

今井らの研究[5]では，重回帰分析とカルマンフィルタを組み合わせて路線バスのバス停に到着する時間を予測する方法を提案している．カルマンフィルタの状態はバス停単位で定められている．また，轟らの研究[6]では，バス停での停車時間，時間帯，天候等を考慮したニューラルネットワークによって路線バスのバス停に到着する時間を予測する方法を提案している．その方法では，バス停出発直後に目的地のバス停の到着時刻を予測するものとなっている．

以上のように，既存研究ではバス停単位の予測がなされている．しかしながら，バス停より細かい粒度で得られるバスの位置を鑑みた遅延状況の推定については議論されていない．また，各研究はそれぞれバス会社から入手したデータを使った検証がなされているが，GTFS のような標準データフォーマットの利用に関する議論もなされていない．

† (株)日立製作所　研究開発グループ
Hitachi, Ltd., R&D Group
E-mail: yoshiyuki.tajima.hh@hitachi.com

3. GTFS データの概要

まず，GTFS で定義される静的データについて簡単に説明する．静的データは表形式のデータで，ファイル名とファイルごとのフィールド名が定められている．ファイルおよびフィールドはそれぞれ必須なものと任意なものに分かれている．任意なものは事業者の判断に応じて記載するかどうかを選択することができる．表 1 に主要なファイル名を示す．GTFS はバスだけでなく鉄道，船舶などでも使用されることが想定されているものであるが，この表はバスを前提に説明を付記している．これらのデータによって経路検索に必要な交通機関，経路，バス停，便，時刻表などの情報を得ることができる．

次に，GTFS-RT で定義される動的データについて簡単に説明する．動的データのデータ構造は木構造となっている．時刻情報のほかに含まれる主要なフィールドを表 2 に示す．この表に示されたフィールドの少なくとも 1 つ以上を提供する必要がある．バスの GTFS-RT データとしては位置を表す VehiclePostion が提供されている場合が多い．本稿では少なくとも VehiclePostion が提供されている場合を想定して検討を進める．GTFS-JP において，更新頻度は 30 秒以内が推奨されている． 実際，宇野自動車株式会社(宇野バス)[7]の GTFS データ[8]の位置情報は 15 秒で更新されていた．

4. 提案手法

4.1 時刻表と遅延の拡張

バスの時刻表は，ある便の停車するバス停の出発時刻が記載されたものを指すことが一般的である．そして，現在時刻が時刻表の出発時刻を過ぎてなお到着しない場合に遅延があるとし，現在時刻からその出発時刻を差し引いた値が遅延時間と呼ばれることが多い．

表 1　GTFS の静的データ(抜粋)

ファイル名	説明
agency	交通機関の名前等
stops	バス停の名前や位置等
routes	経路の名前や説明，輸送手段等
trips	便の名前や説明等
stop_times	バス停の到着時刻と出発時刻
shapes(任意)	詳細な経路中の通過点

表 2　GTFS の動的データのフィールド

フィールド名	説明
TripUpdate	将来/過去の到着/出発時間，予定時間からの遅れ
VehiclePosition	車両の現在位置
Alert	なんらかの事態が発生したことを示す運行情報

本研究では時々刻々と変化するバスの位置に応じて遅延状況を把握するために，これら時刻表と遅延の概念を拡張する．すなわち，路線上の任意の位置でバスが通過すべき時刻が決定的に定められていると仮定する．ここで，通過と表現したのは，到着と出発が同時と見做せることを指した．この仮定の下では，n 次元の実数値空間上で定義される位置を実数値であらわされる通過時刻に変換する通過時刻決定関数$f: \mathbb{R}^n \to \mathbb{R}$が定義できる．そして，現在の時刻$t \in \mathbb{R}$と経路上の位置$x \in X$，$f$が与えられるとき，この位置での遅延時間は$\max(0, t - f(x))$と定義できる．ここで，$X \subset \mathbb{R}^n$は経路上の位置の集合である．位置は路線に複数回同じ場所を通るなどがなければ，GTFS-RT の VihiclePostion 相当でよく，すなわち GPS の緯度と経度の 2 次元でよい．もし部分的な巡回などで同じ場所を通る場合は，それを管理するフラグを含めて位置とすることで対応可能となる．

4.2 時刻表に基づく通過時刻決定関数

前述したとおり，現在の時刻，経路上の位置，通過時刻決定関数が分かれば現在の遅延時間を算出できる．しかしながら，通過時刻決定関数は未知である．本研究では，もとの時刻表の出発時刻に基づいて通過時刻決定関数を定義する．M 個のバス停に関する時刻表を経路上の位置と時刻の対の集合$Q = \{(x_i, t_i) | i = 0, 1, 2, \dots, M-1\}$とする．また，経路上の位置$x$に対して集合$Q$から k 番目に近い位置を取り出す関数を$X_Q(x, k)$，時刻を取り出す関数を$T_Q(x, k)$とする．このとき，経路上の位置$x$での通過時刻決定関数を式(1)とする．

$$f(x) = \alpha T_Q(x, 1) + \beta T_Q(x, 2). \quad (1)$$

ここで，dをユークリッド距離としてαとβは式(2)，式(3)で表される．

$$\alpha = \frac{d\left(x, X_Q(x,2)\right)}{d\left(x, X_Q(x,1)\right) + d\left(x, X_Q(x,2)\right)}, \quad (2)$$

$$\beta = \frac{d(x, X_Q(x,1))}{d\left(x, X_Q(x,1)\right) + d\left(x, X_Q(x,2)\right)}. \quad (3)$$

4.3 遅延時間予測

通過時刻決定関数を導入したことで任意位置での遅延時間が求められる．実用的にはバス停のように定まった位置を指定して固定長化しておくと表示や将来の遅延時間の予測の際に便利である．一方，バスの時速が 40km/h(≒11.1m/s)であった場合，15 秒で 160m 程度は進むこととなる．バス停の間隔が 300~500m と想定すると，位置情報はバス停間の数点程度更新されることになる．これを鑑みると，バス停の総数より数倍程度詳細な経路上の位置であることが期待される shapes を利用することは有用であろう．

そこで本研究では，shpaes に記載される経路上の N 個の位置を活用する．本稿ではこの N 個の位置をチェックポイントと呼び，チェックポイントの順序集合を$X_s = \{x_s^1, x_s^2, \dots, x_s^N\}$とする．添え字について$i < j$ならば$x_s^i$は$x_s^j$より先に到着する位置（$x_s^i \prec x_s^i$）である．$X_s$にはすべてのバス停を含み，$x_s^1$が始点，$x_s^N$が終点のバス停である．一方，チェックポイントに対する通過時刻の順序集合を$T_s = \{t_s^1, t_s^2, \dots, t_s^N\}$とする．添え字について$i < j$ならば$x_s^i$は$x_s^j$より先に到着する位置での通過時刻である． k 番目までのチェックポイントの通過時刻の順序集合をT_s^kとする．同様の考え方で，チェックポイントに対する遅延時間の順序集合を$D_s = \{\delta_s^1, \delta_s^2, \dots, \delta_s^N\}$とする．ここで，任意の i について$\delta_s^i = \max(0, t_s^i - f(x_s^i))$である．また，k 番目までのチェックポイントの遅延時間の順序集合をD_s^k，k+1 番目以降のチェックポイントの遅延時間の順序集合を$D_s^{\tilde{k}}$とする．以上の定義のもと本研究では，D_s^kが与えられたときに$D_s^{\tilde{k}}$を推定すること遅延時間予測と呼ぶ．

D_s^kや$D_s^{\tilde{k}}$の長さは時々刻々と変化する．異なる長さのD_s^kや$D_s^{\tilde{k}}$に応じて予測モデルを構築するとモデル管理が煩雑となる．そこで，予測モデルの入出力の長さを N に揃える．具体的には，D_s^kの k+1 番目以降に-1 などのダミーの値を割り当てて長さをそろえる．同様に，予測する際は，k+1 番目から予測するのではなく，1 番目から N 番目までをすべて予測して k+1 番名以降のみを使う．

5. GTFS データを用いた実験

5.1 実験の概要

提案手法の実現性と効果を確認するため，宇野バスの GTFS データを活用して遅延状況の可視化を行った．また，遅延時間予測をバス停単位とした場合と，チェックポイント単位とした場合の予測性能の違いを比較した．

5.2 収集した静的データ

宇野バスから収集した GTFS データの例を示す．表 3 は stops データを抜粋したものである．バス停の識別子，名前，緯度経度などが記載されている． 表 4 は trips データを抜粋したものである．便の識別子(tirp_id)と走行経路の shape に関する識別子(shape_id)が紐づけられている．表 5 は stop_times データを抜粋したものである．バス停の到着および出発時刻が記載されている．表 6 は shape データを抜粋したものである．shape_id に対するチェックポイントの緯度経度が記載されている．

表 3　stops データの例(抜粋)

stop_id	stop_name	stop_desc	stop_lat	stop_lon
2_親	岡山駅	親（駅・ターミナル）	34.66498	133.9186
2_01	岡山駅		34.66495	133.9186
2_04	岡山駅		34.66509	133.9187
3_03	柳川西（北側）東岡山		34.66571	133.9239
4_親	岡山駅前・ドレミの街	親（駅・ターミナル）	34.66569	133.9213

表 4　trips データの例(抜粋)

route_id	service_id	trip_id	trip_headsign	shape_id
ネオ瀬戸線_A	平日	平日_05 時 40 分_系統 1642	瀬戸駅（岡山駅経由）	1642
林野線_A	平日	平日_06 時 00 分_系統 1332	表町 BC（新道河本	1332
林野線_A	平日	平日_06 時 00 分_系統 1182	表町 BC（新道河本	1182
ネオポリス線_A	平日	平日_06 時 05 分_系統 1282	表町 BC（下市・中	1282
ネオポリス線_A	平日	平日_06 時 10 分_系統 1632	表町 BC（下市・西	1632

表 5　stop_times データの例(抜粋)

trip_id	arrival_time	departure_time	stop_id
平日_05 時 40 分 _系統 1642	5:40:00	5:40:00	792_05
平日_05 時 40 分 _系統 1642	5:40:00	5:40:00	790_05
平日_05 時 40 分 _系統 1642	5:40:00	5:40:00	788_05
平日_05 時 40 分 _系統 1642	5:40:00	5:40:00	786_05
平日_05 時 40 分 _系統 1642	5:41:00	5:41:00	784_05

表 6　shape データの例(抜粋)

shape_id	shape_pt_lat	shape_pt_lon	shape_pt_sequence
1041	34.66495	133.9186	1
1041	34.66469	133.9184	2
1041	34.66477	133.9182	3
1041	34.66563	133.9188	4
1041	34.66569	133.9213	5

5.3 運行状況の可視化

GTFS データ（静的データ，動的データ）を活用して運行状況の可視化を行った．図 1 は，2021 年 1 月 26 日に収集した，ネオ瀬戸線，岡山駅発瀬戸駅行きである系統 1061, 7 時 50 分発のバスの位置データを地図上にプロットしたものである．この路線は約 1 時間で走行される．図 1 の例では重複を除くと 93 点の位置がプロットされている．

2021 年 1 月 25 日～29 日における系統 1061，7 時 50 分発のバスの遅延時間の推移を図 2 に示す．上から 25 日，26 日，27 日，28 日，29 日のものである．横軸はチェックポイントの番号である．この路線のチェックポイントは合計で 163 点となっている．縦軸は遅延時間であり，単位は分である．遅延時間の計算には前述の通過時刻決定関数を用いている．

図 2 より日別の遅延状況の違いが可視化できている．具体的には，1 月 25 日から 28 日までは長くとも 10 分程度の遅延となっている．特に 1 月 28 日は遅延が比較的少ない．これに対し，金曜日では 15 分程度の遅延となっていることがわかる．

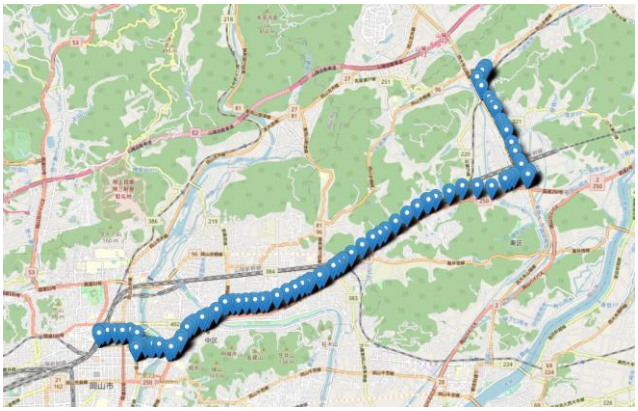

図 1　2021 年 1 月 26 日に収集した系統 1061
(岡山駅→瀬戸駅) 7 時 50 分発のバスの位置データの軌跡

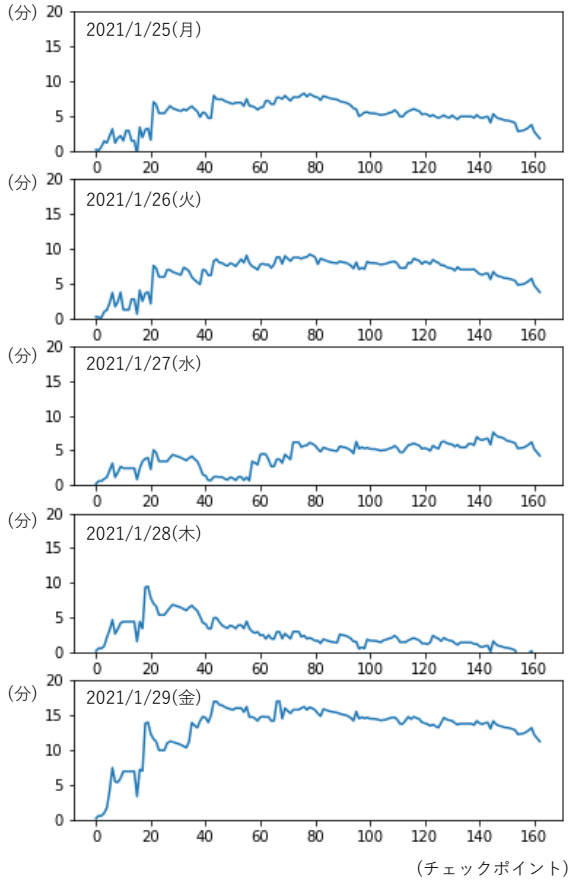

図 2　2021 年 1 月 25 日～28 日における系統 1061
(岡山駅→瀬戸駅) 7 時 50 分発のバスの遅延時間の推移

5.4 粒度を変えた遅延時間予測に関する実験

提案した方法によって駅単位より細かいチェックポイント単位で予測での予測が可能となる．この粒度の違いが遅延時間予測の予測性能に影響を与えるかどうかを確かめるために実験を行った．なお，前述のとおりD_s^kや$D_s^{\bar{k}}$の長さをそろえることで時々刻々と変化する予測におけるモデルを共通化する方法を提案した．一方，本実験では粒度の違いによる予測性能に着目している．そのため，対象とした便に関して，最初の 7 割のチェックポイントをD_s^k，残りの 3 割のチェックポイントを$D_s^{\bar{k}}$とした予測モデルを構築するものとした．

実験には系統 1061 の平日の GTFS データを用いた．前述のとおり系統 1061 の場合チェックポイントは合計 163 点ある．そこで，D_s^kを最初の 122 点，$D_s^{\bar{k}}$を残りの 41 点とした．データは，2021 年 1 月 25 日から 2 月 26 日までを訓練用，3 月 1 日から 3 月 31 日までをテスト用とした．バス停単位の予測モデルの訓練ではバス停となっているチェックポイントの情報のみを使うものとした．これに対し，チェックポイント単位の予測モデルの訓練では，すべてのチェックポイントの情報を使うものとした．一方，テストではバス停となっているチェックポイントを用いた．すなわち，チェックポイント単位の予測モデルでは，一旦すべて

のチェックポイントに関する予測を行った上でバス停に関するチェックポイントの結果だけをテストに用いた．

予測モデルの実装には scikit-learn[9]に実装されている RandomForestRegressor を用いた．実験では異なるシードで 10 回試行して得られた Mean Absolute Error(MAE)の平均を評価指標とした．ハイパーパラメータは次のようにした．大きいほど良いとされる木の数(n_estimators)は初期値である 100 から 500 に変更した．木の深さは 3 から 8 まで変化させた．それ以外はライブラリの初期値を用いた．

実験の結果を表 7 に示す． 木の深さによらずチェックポイント単位の予測モデルがバス停単位の予測モデルより小さい MAE となった．したがって，駅単位からチェックポイント単位に粒度を細かくすることは予測結果の粒度を細かくするだけでなく，予測モデルの性能向上にも寄与したといえる．

5.5 考察

提案手法によってチェックポイント単位の遅延の可視化ならびに予測が可能となった．これは駅単位に比べてユーザに多くの情報を与えることができる点で効果的である．また，GTFS に準拠していれば本技術を導入することができるので汎用的であることが期待できる．

また，宇野バスの GTFS データを使った実験から，バス停単位で予測を行うよりもチェックポイント単位で予測を行う方が性能向上を見込めることが明らかとなった．冒頭で述べた MaaS の実現に向けて必要な性能を達成しているかどうかは明らかではないが好材料といえる．

6. おわりに

本研究では路線バスの GTFS データを使って時々刻々と変化する遅延状況を推定する方法を提案した．特に，GTFS の shapes に記載された位置の活用，通過時刻決定関数の導入によって，駅単位より細かい粒度であるチェックポイント単位での遅延状況を推定することを可能とした．加えて，チェックポイント単位で予測モデルを構築することで予測性能を向上できることを宇野バスの GTFS データを使って示した．

今後の課題としては，より正確な予測を実現するために，既存研究でも取り組まれているように，天候や曜日など遅延に影響を及ぼすほかの要因を考慮することが挙げられる．また，ほかのバス会社の GTFS データを使って本技術の汎用性を具体検証することも挙げられる．

表 7　バス停単位の予測モデルとチェックポイント単位の予測モデルの比較結果

木の深さ	バス停単位の MAE の平均(分)	チェックポイント単位の MAE の平均(分)
3	0.6750	0.6469
4	0.6428	0.6264
5	0.6333	0.6182
6	0.6304	0.6161
7	0.6285	0.6153
8	0.6291	0.6150
平均	0.6398	0.6230

参考文献

[1] 国土交通省，“MaaS（モビリティ・アズ・ア・サービス) について”，https://www.mlit.go.jp/pri/kikanshi/pdf/2018/69_1.pdf
[2] Google，“General Transit Feed Specification（GTFS）”，https://developers.google.com/transit/gtfs/ .
[3] Google，“GTFS Realtime”，https://developers.google.com/transit/gtfs-realtime .
[4] 標準的なバス情報フォーマット広め隊，“標準的なバス情報フォーマット”，https://www.gtfs.jp/ .
[5] 今井 瞳，廣井 慧，河口 信夫，“複数事業者の路線バス運行実績データに基づく到着時刻予測モデルの提案と精度検証”，研究報告高度交通システムとスマートコミュニティ（ITS），Vol. 23(2017).
[6] 轟朝幸，川崎智也，野村大智，横関敬裕，“ニューラルネットワークを用いた路線バスの遅延時間予測”，交通工学論文集，Vol. 3, No.2, pp. A_202-A_207(2007).
[7] 宇野バス(宇野自動車株式会社)，https://www.unobus.co.jp/ .
[8] 宇野バスオープンデータ，http://www3.unobus.co.jp/opendata/ .
[9] scikit-learn，https://scikit-learn.org/ .