バス時刻表の最適化

高松　瑞代

私たちが普段利用している電車やバスの時刻表は，かなり工夫して設計されている．たとえば，大きな駅では電車とバスがスムーズに乗換できるように時刻表が組まれている．また，学校の前にあるバス停には朝の授業が始まる少し前にバスが着くようになっているだろう．それにもかかわらず，利用者から見るともっと便利になればよいのにと思うことがある．現在の時刻表をさらに改善するためには，どのような点に気をつけて時刻表を設計すればよいのだろうか．本稿では，最適化手法を用いてバス時刻表を設計する方法を紹介する．

キーワード：時刻表設計，最適化，ネットワーク，モデリング

1. はじめに

電車とバスを乗り継ごうとして駅に着いたらバスがすでに出発していた，という経験はないだろうか．そのようなとき，多くの人は「電車を降りてすぐにバスに乗れないなんて，不便な時刻表だな」と感じるだろう．本稿では，最適化という手法を使って時刻表の不便さを解消する方法を紹介する．

対象とするのは，鉄道・バスが低頻度で運行されている岩手県の地域である（図 1）．表 1 は世田米駅前というバス停の時刻表である．表 1 を見ると，このバス停では各行き先に対して一日に 3 本または 4 本しかバスがないことがわかる．

表 1 のバス停の近くに駅があり，7:50 に出発する電車に乗り換えたいとする．7:46 発の盛岡行きのバスで世田米駅前のバス停に着いた乗客は，7:50 の電車に乗ることができる．一方，7:50 の電車で駅に着いた乗客が盛岡行きのバスに乗ろうとする場合，12:46 まで 5 時間近く待たなければならない．鉄道とバスの運行頻度が低い地域では乗換の待ち時間が長くなり，住民のバス利用を遠ざける要因となっている．したがって，このような地域における「便利な時刻表」とは，「利用者が円滑に乗り換えられる時刻表」と言い換えることができる．

もちろん，バスの運行本数を増やせば乗換の待ち時間は短くなる．しかし，運行本数を増やすとバスの台数や運転手の人数も増やさなければならないため，多くの費用がかかる．そのため，バス会社としては運行本数の増加などの大きな変更は避けたい．本稿では，バスの運行本数は変更せず，バスの発着時刻をずらすことで円滑な乗換を実現する．このようなちょっとした工夫で，設備投資や人件費に負担をかけることなく時刻表が便利になることを示す．

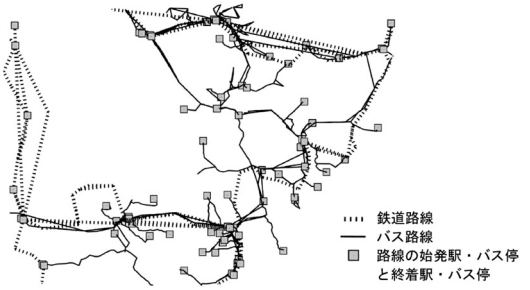

図 1　対象地域の鉄道およびバス路線

表 1　岩手県にあるバス停「世田米駅前」の時刻表（2012 年 2 月 14 日当時）

盛岡行き	中井行き	住田高校行き
06:46	08:18	07:27
07:46	13:08	11:17
12:46	16:28	16:17
16:46	18:43	

2. 最適化とは

前節で述べた問題設定をまとめると以下のようになる．

目的　利用者の乗換を円滑にする

制約　現在の時刻表を大幅に変更することはしない（具体的には，バスの本数は変更せず，バスの運行順序はできる限り現状を維持する）

最適化とは，制約を満たすものの中でもっともよい目的を達成することである．最適化の問題は，高校数

たかまつ　みずよ
中央大学理工学部情報工学科
〒 112–8551 東京都文京区春日 1–13–27
takamatsu@ise.chuo-u.ac.jp

学の中にもたくさん見つけることができる．

例題 1. 2 次関数 $f(x) = x^2 - 2x + 3$ を $0 \leq x \leq 5$ において最小にする x とその最小値を求めよ．

例題 1 の目的は $f(x) = x^2 - 2x + 3$ の値をできるだけ小さく（最小化）すること，制約は $0 \leq x \leq 5$ である．次に，もう少し難しい最適化問題の例を見てみよう．

例題 2. $x \geq 0, y \geq 0, x + y \leq 4, 3x + y \leq 6$ で表される領域内で，関数 $g(x, y) = 5x + 2y$ を最大にする点とその最大値を求めよ．

例題 2 の目的は $g(x, y) = 5x + 2y$ の値をできるだけ大きく（最大化）すること，制約は $x \geq 0, y \geq 0, x + y \leq 4, 3x + y \leq 6$ である．例題 1 の変数は x のみであり，例題 2 の変数は x，y の二つである．これらの例題は手計算でも解くことができる[1]．

本稿で扱うバス時刻表設計問題についても，先に述べた目的と制約を数式で記述することにより数学の問題に帰着できる．ただし，変数の数は数万個になる．このように，現実に現れる最適化問題は数万や数十万といった非常に多くの変数をもつ．

多くの変数をもつ最適化問題はどのようにして解けばよいだろうか．すぐに思いつく手段は，コンピュータを使うというものである．しかし実際には，コンピュータを使っても，もっともよい目的を達成する解（最適解）を計算することが難しい場合がたくさんある．

たとえば，半年後までにバス時刻表を提案しなければならない状況において，「最適解を計算するのに 5 年かかる」と言われても困るだろう．最適化問題として記述する際には，自分が解きたい現実の問題をどのようにして実用的な計算時間で解ける問題にするか（モデリング）が非常に重要である．オペレーションズ・リサーチの研究は，モデリングから始まると言っても過言ではない．文献 [1] では，オペレーションズ・リサーチの研究者がさまざまな立場からモデリングについて論じている．

本稿では，「バス時刻表を便利にする」という問題を「約 4 秒で計算できる最適化問題」として表現する方法を説明する．本稿の内容は [2, 3] に基づいている．

[1] 例題 1 は $x = 1$ のとき最小値 2，例題 2 は $x = 1, y = 3$ のとき最大値 11 となる．

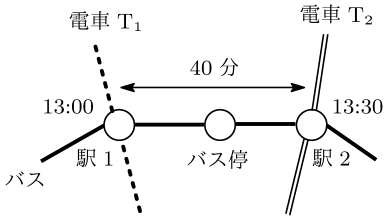

図 2 相互乗換の両立が難しい例

3. 時刻表の最適化

3.1 乗換を円滑にするために

1 節で述べたように，バスの運行本数が少ない地域では次のバスが到着するまで数時間かかる．通常，日本ではバスや電車の停車時間が短い．そのため，バス X から電車 Y に乗換可能であるときに，電車 Y が到着する前にバス X が出発してしまい，電車 Y からバス X に乗り換えることはできない．これは，鉄道・バスの運行本数が少ない地域では致命的である．乗換を円滑にするためには，バス X から電車 Y に乗換可能ならば，電車 Y からバス X にも乗換可能であることが望ましい．このような乗換を相互乗換と呼ぶ．以下では，「乗換を円滑にする」という目的を「できるだけ多くの相互乗換を実現する」と言い換える．

表 1 の例では，バスの到着時刻と出発時刻が同じであり，停車時間は 0 分である．もし 7:46 にバス停に到着するバスを 7:55 までバス停に待機させると

- 7:46 到着のバスから 7:50 出発の電車への乗換
- 7:50 到着の電車から 7:55 出発のバスへの乗換

が両方とも可能になる．しかし，この場合はバス停に 9 分間停車しており，停車時間が長くなるという問題が生じる．

相互乗換を実現すると，乗換客がバス停で待つ時間が短くなる．一方，バスの停車時間が長くなると，バスに乗っている乗客がバスの中にいる時間が増加する．このように，相互乗換を実現することとバスの停車時間を長くすることはトレードオフの関係になっている．したがって，バスの停車時間が長くなりすぎないように工夫して相互乗換を増やさなければならない．

相互乗換を「いつ・どこで」実現するかを決めるのは難しい問題である．例として，図 2 に示すように，駅 1 と駅 2 にあるバス停を通るバスを考えよう．電車 T_1 は 13:00 に駅 1 に到着し，電車 T_2 は 13:30 に駅 2 に到着する．また，バスが駅 1 から駅 2 まで走行するのに 40 分かかる．いま，電車の時刻表に合わせてバスの時刻表を設計したいとする．電車 T_1 との相互

表 2　バス B の時刻表

バス停	1 始発	2 世田米駅前	3	4 終点
到着時刻	—	07:46	08:05	08:35
出発時刻	07:30	07:46	08:10	—

表 3　バス B の時刻表に対する変数

バス停	1 始発	2 世田米駅前	3	4 終点
到着時刻	—	x_2	x_3	x_4
出発時刻	y_1	y_2	y_3	—

乗換を可能にすると，バスは駅 1 を 13:00 過ぎに出発する．その結果，駅 2 に到着するのは 13:40 過ぎになり，電車 T_2 との相互乗換が不可能になる．同様に，電車 T_2 に合わせてバスの時刻表を決めると，今度は電車 T_1 との相互乗換が不可能になる．

図 2 のように，バス路線や鉄道路線では複数のバス停や駅がつながっており，互いに引っ張り合っているような状況にある．そのため，ある場所での乗換を部分的に改善しても，他の場所での乗換が悪化することがある．相互乗換を「いつ・どこで」実現するかを決めるためには，バス路線と鉄道路線の引っ張り合いの関係を考慮し，それぞれの場所での乗換が他の場所にどのような影響を及ぼすかを把握する必要がある．本稿では最適化問題を解くことで，相互乗換を実現する箇所を決定する．

3.2　相互乗換に関する制約を記述する

最適化問題の制約を数式で記述する．世田米駅前を 7:46 に発着するバス B が，表 2 にある時刻表のとおりに四つのバス停 1，2（世田米駅前），3，4 を通るとする．以下，円滑な乗換を保証する時刻表を設計するために，バスの到着時刻と出発時刻を変更する．

表 3 に示すように，バス B がバス停 i に到着する時刻を x_i，バス停 i を出発する時刻を y_i とする．ここでは，バス停 2（世田米駅前）における相互乗換を考えるときに，変数 x_2，y_2 が満たすべき条件を記述する．

変更後の時刻表において，バス B から 7:50 発の電車への乗換を維持するためには

（バスの到着時刻）$\leq$（電車の出発時刻），

つまり

$$x_2 \leq 7{:}50 \tag{1}$$

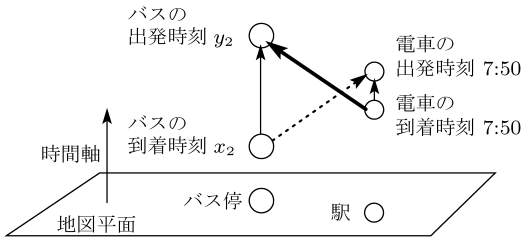

図 3　相互乗換の例

が成立しなければならない[2]．簡単のため，ここでは乗換にかかる時間を 0 分としている．

バス停 2 で相互乗換を実現するか否かは，他のバス停との関係まで考慮して決めなければならない．そこで，相互乗換を実現するか否かを表す変数 z を用意する．変数 z は 0 または 1 の値をとり，

- $z = 1$ ならば，相互乗換を実現する
- $z = 0$ ならば，相互乗換を実現しない

ことを意味する．もし $z = 1$ ならば，現在の時刻表では不可能な乗換（電車→バス）が可能になるので，

（バスの出発時刻）$\geq$（電車の到着時刻），

つまり

$$y_2 \geq 7{:}50 \tag{2}$$

が成立する．図 3 は相互乗換を表している．図 3 において，実線はバス・電車の停車，点線はバスから電車への乗換，太線は電車からバスへの乗換を意味する．一方，$z = 0$ ならばこの乗換は不可能なままでよいので，(2) を満たす必要はない．

これらの条件は，次の一つの式で表現することができる．

$$y_2 - 7{:}50 + 10000(1 - z) \geq 0 \tag{3}$$

$z = 1$ ならば (3) は (2) と一致し，$z = 0$ ならば (3) の左辺には 10000 が足されるので，y_2 が 7:50 より早い時刻であっても常に 0 以上になる．変数 z は，相互乗換を実現するか否かで式の意味を変えるために使われている．0 または 1 という 2 通りの値しかとらない変数 z を利用すると，上記のような複雑な制約を記述することができる．(3) では 10000 を使用したが，十分大きい数ならば他の数でも問題ない．このように，大きな定数 (big-M) を用いる定式化の手法を big-M 法と呼ぶ．

[2] 実際には，変数 x_2，y_2 の単位を分として，時刻 7:50 を $7 \times 60 + 50 = 470$ [分] に直した式を考える．

3.3 時刻表の制約を記述する

次に，表 3 の変数 y_1，x_2，y_2，x_3，y_3，x_4 がバスの到着・出発時刻であるために満たすべき条件を考える．各バス停において，出発時刻は到着時刻以上でなければならないので，

$$y_2 \geq x_2, \quad y_3 \geq x_3 \tag{4}$$

が成り立つ．

時刻表を変更してもバス停間の走行時間は同じである．表 2 をみるとバス停 1 からバス停 2 までの走行に 16 分（7:30 と 7:46 の差）かかるため，変更後の時刻表においても

$$x_2 - y_1 = 16 \text{ [分]} \tag{5}$$

が成り立たなければならない．同様にして，

$$x_3 - y_2 = 19 \text{ [分]}, \quad x_4 - y_3 = 25 \text{ [分]}$$

も必要である．

これらは一つのバス B に関する制約であるが，同じ路線の他のバスに対する先発・後発の関係も考えなければならない．バス B′ がバス B のあとに運行しているとする．バス B′ がバス停 i を出発する時刻を y_i' とおくと，バス B′ とバス B の先発・後発の関係を維持するためには

$$y_1 \leq y_1', \quad y_2 \leq y_2', \quad y_3 \leq y_3' \tag{6}$$

を考える必要がある．

3.4 目的を記述する

相互乗換を実現するために，式 (4) はバス停での停車時間（たとえば $y_2 - x_2$）が現在の時刻表よりも長くなることを許している．そのため，今まで述べた制約だけでは，バス停での停車時間が長くなりすぎる危険性がある．

相反する二つの目標

- 相互乗換を増やしたい
- バス停での停車時間の延長をできるだけ避けたい

のトレードオフの関係を表現するために，

（相互乗換の導入により減少する移動時間の総和）
$$-\text{（バス停での停車時間の総和）} \tag{7}$$

を最大化することを考える．この目的を採用することで，バス停での停車時間を無駄に長くすることなく，多くの相互乗換を実現することができる．詳細は省略するが，(7) は x_i，y_i，z に対応する変数の一次式で記述することができる．

3.5 最適化問題のまとめ

提案する最適化問題は，次の 3 種類の変数をもつ．

- 各バスのバス停への到着時刻 x_i
- 各バスのバス停からの出発時刻 y_i
- 相互乗換を実現するか否かを表す変数 z

変数 x_i，y_i は 0 時から 24 時までの時刻を表す．この制約は

$$0{:}00 \leq x_i \leq 24{:}00, \quad 0{:}00 \leq y_i \leq 24{:}00$$

で記述することができる．一方，z は 0 または 1 の値しかとることができないことに注意する．3.2 節と 3.3 節で述べた制約をまとめると以下のようになる．

- 現在の乗換を維持する式 (1)
- 相互乗換に関する式 (3)
- バス停における出発・到着時刻に関する式 (4)
- バス停間の走行時間に関する式 (5)
- 同じ路線のバスの先発・後発を維持する式 (6)

これらの制約のもとで，式 (7) の値を最大化する．

以上で，最適化問題の目的と制約を記述することができた[3]．この問題では，実数値をとる変数 x_i，y_i と 0 または 1 である変数 z が混在している．また，目的と制約はすべて x_i，y_i，z に関する一次式で記述されており，x_i^2 や $y_i z$ などの項は存在しない．このような最適化問題を混合整数計画問題という．

混合整数計画問題は，ソフトウェアを使って手軽に解くことができる [4]．図 1 の地域については，x_i，y_i に対応する変数が約 6 万個，z に対応する変数が約 4 千個，制約の式が約 11 万本と大規模になるが，CPLEX という整数計画ソルバー（ソフトウェア）を用いて約 4 秒で最適解を計算することができる．

3.6 時刻表をネットワークで表現する

本研究は

- 現在の時刻表が不便である原因を見つける
- 不便さを解消する問題を最適化問題として記述する

の二段階から成り立っている．前半では，現在の時刻表を利用した場合の移動時間を計算することで，乗換時間が長いことが不便さの原因であることを明らかにした．後半では相互乗換に着目し，多くの場所で相互乗換を実現する時刻表を設計する問題を混合整数計画問題として記述した．どちらの段階においても，時刻表どおりに乗客の移動を表現できる時空間ネットワークが基盤となっている．

[3] 詳細は [3] に記載されている．

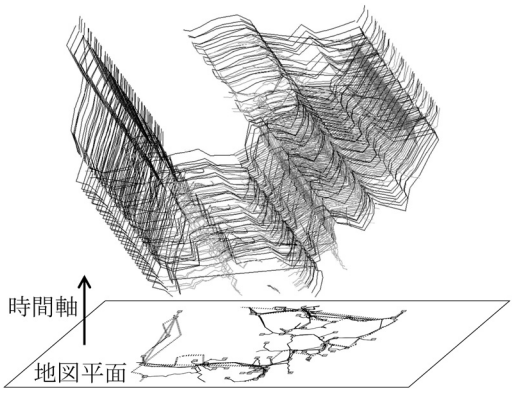

図 4　図 1 の地域に対する時空間ネットワーク

表 4　各時間帯に出発して 9 時に病院へ到着するバス停数の比較

	7:00 ～ 7:30	7:30 ～ 8:00	8:00 ～ 8:30	8:30 ～ 9:00	合計
現行	244	196	114	98	652
改訂	303	234	120	99	756

時空間ネットワークは，各バスおよび電車の発着に対応する頂点と，バスおよび電車の移動を表す枝からなるネットワークである [5]. 時空間ネットワークの規模はもとの路線図のネットワークに比べて格段に大きくなるが，時間に依存した乗客の動きを正確に表現できるという大きな特長がある．

図 4 は，図 1 の路線図に対して構築した時空間ネットワークである．頂点の数は約 6 万，枝の数は約 11 万という非常に大規模なネットワークになっている．

3.7　提案する時刻表と現行時刻表の比較

図 1 の地域に対して最適化問題を解いて求めた改訂時刻表を，現在の時刻表と比較する．表 4 は，対象地域にある病院に 9 時までに着くことを目標としたとき，各時間帯に出発すればよい地点の数を表す．表 4 を見ると，すべての時間帯で改善されていることがわかる．

改訂時刻表ではバスの停車時間の約 97%が現行時刻表と同じであり，バスを無駄に長く停車させることなく相互乗換を実現することに成功している．

同様の手法によって得られた改訂時刻表と現行時刻表を比較する動画が [6] にて公開されている．この動画では改訂時刻表と現行時刻表の「追いかけっこ」を見ることができ，二つの時刻表の違いをより深く実感することができる．

4.　おわりに

本稿では，バスの時刻表を設計する問題を最適化問題として記述して解く手法を紹介した．バスの時刻表という非常に身近なものに関する問題を，相互乗換に着目してモデリングを行うことで，中学校で習うような一次式を使って表現している．一本一本の式は難しくないが，本稿で紹介した最適化問題は 10 万本以上の式から成り立っている．このように，現実に現れる最適化問題は非常に大規模であるが，近年の最適化アルゴリズムの発展により整数計画ソルバーを用いるとたった数秒で解くことができる．

ヨーロッパでは鉄道最適化という分野が盛んに研究されており，最適化手法を用いて設計された鉄道時刻表が実際に利用されている．最適化は，数学とコンピュータを使って社会の問題を解決できる非常におもしろい分野である．本稿を通じて最適化の威力を感じていただけたら嬉しく思う．

謝辞　本稿の執筆の機会を与えてくださり，多くの有益なコメントをくださった東京農工大学の宮代隆平先生に深く感謝いたします．また，本稿の執筆にあたり貴重なご意見をくださった中央大学の田口東先生に感謝いたします．

参考文献

[1] 室田一雄，池上敦子，土谷隆（編），赤池弘次，伊理正夫，茨木俊秀，腰塚武志，小島政和，福島雅夫，森戸晋，逆瀬川浩孝，木村英紀，深谷賢治，鈴木敦夫，藤原祥裕，田村明久，久保幹雄，松井知己（著），『モデリング—広い視野を求めて—』，近代科学社，2015.

[2] 赤星健太郎，高松瑞代，田口東，石井儀光，小坂知義，“低頻度な公共交通網を有する地域の移動利便性の評価手法に関する研究—時空間ネットワークを用いた公共交通網と都市構造の関連分析—，”都市計画論文集，**47**, pp. 847–852, 2012.

[3] M. Takamatsu and A. Taguchi, “Train and bus timetable design to ensure smooth transfer in areas with low-frequency public transportation services,” In *Proceedings of the 6th International Conference on Railway Operations Modelling and Analysis (Rail-Tokyo2015)*, Paper ID 34, 2015.

[4] 宮代隆平，“整数計画ソルバー入門，”オペレーションズ・リサーチ：経営の科学，**57**, pp. 183–189, 2012.

[5] 田口東，“首都圏電車ネットワークに対する時間依存通勤交通配分モデル，”日本オペレーションズ・リサーチ学会和文論文誌，**48**, pp. 85–108, 2005.

[6] 汎用可視化ソフトウェア AVS の適用事例，http://www.cybernet.co.jp/avs/example/category/transport.html

