中山間地におけるデマンドバスシステムの開発
-デマンドバスの実際的経路探索法-

岡谷 正博　　平田 富夫
名古屋大学大学院工学研究科
〒464-8603 名古屋市千種区不老町
Tel :052-789-3440, Fax :052-789-3089
E-mail: {okaya,hirata}@hirata.nuee.nagoya-u.ac.jp

誉田 安秀
(株) 松下通信工業
〒223-8639 横浜市北区綱島 4-3-1
Tel :045-544-3395, Fax :045-544-3414
E-mail: konda@isd.mci.mei.co.jp

概要　ITS の目的のひとつに乗客輸送における利便性の向上と, 事業の効率化がある. その具体化の一環として, 利用者の乗降車要求によりバスの運行計画 (運行時間, 運行経路) を変更するデマンドバスが提案されている. 我々は, バスの利用頻度の低い中山間地において, バスの運行経路を変更できるデマンドバスシステムを構築した. このデマンドバスシステムではバスの運行計画をソフトウェアによりリアルタイムで作成する. このソフトウェアはデマンドエンジンと呼ばれ, タブー探索を用いて高速に運行コスト最小の運行計画を見つける. 本研究では, まず, デマンドエンジンが出力する運行計画がほぼ最適であることを確認する. 次に, タブー探索アルゴリズムの重み係数の最適な値を求める. さらに, バスを２台に増やした場合の性能評価を行う.

Development of Demand Responsive Bus System in Rural Areas
-Practical Routing Algorithm -

Masahiro Okaya　Tomio Hirata
Graduate School of Engineering, Nagoya University
Furo-cho Chikusa-ku Nagoya, 464-8603
Tel :+81-52-789-3440, Fax :+81-52-789-3089
E-mail: {okaya,hirata}@hirata.nuee.nagoya-u.ac.jp

Yasuhide Konda
Matsushita Communication Industrial Co.,Ltd.
4-3-1, Tsunasima Kita-ku Yokohama , 223-8639
Tel :+81-45-544-3395, Fax :+81-45-544-3414
E-mail: konda@isd.mci.mei.co.jp

Abstract　Improvement of user's convenience in the public transportation and efficiency of its business administration is one of the purposes of ITS. Demand responsive bus system has been proposed for this purpose. We have launched a demand responsive bus system in a rural area in Japan . This system incorporates software, called Demand Engine, which produces bus routes responding user's requests quickly. The demand engine exploits tabu search for finding an optimal bus route. In this research, we first verify that this engine finds an almost optimal solution efficiently. Next, we tune up a parameter of the engine so that it produces better solution. Finally, we examine if it can be extended for treating multiple vehicles.

1 はじめに

近年, 高度道路交通システム (ITS) についての研究開発が活発に行われている. ITS は道路交通を高度に情報化, インテリジェント化することにより, 道路交通の円滑化, 環境の改善を目指すものである. その一手法として, 乗客輸送における効率化を図る目的で様々な乗合バスの方式が提案されており, そのひとつにデマンドバスシステムがある.

デマンドバスシステムは, 欧米では２０年程前から実験が行われているが, 当時の技術ではまだ人手に負うところが多く, 効率的なシステムを構築するには至らなかった. また, 国内では路線の許認可の問題もあり, 一部の運行経路に限り, 要求があれば迂回をするというような簡単な方式に留まっていた. しかし, 今日では通信・情報技術の発達とともに, デマンドバスシステムの実用化が現実のものとなった.

過疎化が進んだ地域では, バス利用者が激減し, バスの路線によっては収益が下がり存続が危ぶまれるものがある. その解決策としてデマンドバスシステムが注目を浴びている. 高知県中村市では, 南北約３キロ, 東西約４キロの地域におよそ６０箇所の停留所を用意し, 国内では初めてのフルデマンド方式のデマンドバスが 2000 年４月より実験的に運行され成功を収めた.[6]

このデマンドバスシステムでは, バスの運行計画をソフトウェア（デマンドエンジン）によりリアルタイムで作成している. デマンドエンジンは, 利用者の乗車希望時間, 乗降車希望場所に対して運行コスト最小の運行計画を作成する.

本論文の目的は, デマンドエンジンの性能評価である. 最初に, デマンドエンジンが利用者の乗降車要求に対して, ほぼ最適な運行計画を見つけていることを示す. 次に, タブー探索の効果を確認し, タブースパンの最適な値を見つける. 最後に, バスを２台に増やした場合の性能を評価する.

以下, ２節でデマンドバスシステムの概要を, ３節で経路探索の実際的計算手法を述べる. ４節で計算機実験の結果を示す. ５節はまとめである.

2 デマンドバスシステムの概要

2.1 運行コスト

運行計画は, バス走行の効率性と利用者の利便性の両者を満足させるようにしたい. 具体的には, バスの走行時間, 利用者の乗車時間と乗降車点間の最短移動時間との差, および, 利用者の乗降車希望時間とバスの到着時間との差である. これらの時間に後で定義する利用者の属性に応じた重み係数を乗じる. この総和を運行コストと呼び, バスの運行計画の評価基準とする. デマンドエンジンは運行コストの最小化を目的としている. 以下, 運行コストに関する定義を与える.

1. バス b_j の走行時間を run_j とする.run_j に対する重み係数を α_j とする.
2. 利用者 u_i に対して, u_i のバスの乗車時間と u_i の乗降車点間の最短移動時間との差を $slack_i$ とする. つまり,$slack_i$ は利用者 i にとっての余分な乗車時間である. $slack_i$ に対する重み係数を β_i とする.
3. 利用者 u_i に対して, 乗車希望時間, 降車時刻（特急列車利用者のみ）からのバスの遅延時間を $delay_i$ とする. （ただし, 乗車希望時間に対しては数分の許容時間 t^a を設けてその時間以内の遅延は許容している.）$delay_i$ に対する重み係数を γ_i とする.
4. 利用者 u_i の乗車希望時間に対するバスの早着時間を $early_i$ とする. $early_i$ に対する重み係数を δ_i とする.

各時間に重み係数を乗じたものの総和が運行コストである. 重み係数 β, γ, δ は利用者の属性により変えることができることに注意されたい. これにより, 利用者の多様な乗降車要求に対応する運行計画を見つけることが可能となる. 我々の目標は, 下に示す目的関数の値が最小となるバスの運行計画を求めることである.

$$\text{目的関数 } F = min\{\sum \alpha_j \times run_j + \sum \beta_i \times slack_i + \sum \gamma_i \times delay_i + \sum \delta_i \times early_i\}$$

2.2 乗降車要求

利用者の乗車希望時間と乗降車地点を乗降車要求と呼ぶ. 本システムでは, 利用者の乗降車要求に対し, デマンドエンジンが最適な運行計画を作成しバスの運行を開始するが, バス運行中に新たな乗降車要求が発生すると, 予定の運行経路を変更するようにバスに無線で知らせることが出来る（これをフルデマンドバス方式と呼ぶ）. 乗降車要求を予約時期によって２つに分ける. バスが出

発する以前に分かっているもの（通常の乗降車要求）と，バスが出発した後に発生するもの（オンライン乗降車要求）である．

オンライン乗降車要求が発生した場合，現在の運行計画を破棄し，まだサービスが終了していない利用者（その時にバス乗車中の利用者とこれから乗車を予定している利用者）と新たに加わった利用者を一緒にして運行計画を再構築する．オンライン乗降車要求に対しては，通常の乗降車要求より重み係数 β, γ, δ を小さくする．つまり，$slack, delay, early$ の値が同じでも運行コストは通常の乗降車要求の方が大きくなる．したがって，エンジンは運行コストを下げるために，バス運行開始以前に乗車予約をした利用者をオンライン乗降車要求の利用者よりも優先する運行計画を作成する．

また，特急列車へ乗継ぐ利用者は駅への到着時間を遅らせることはできない．この為，特急利用者の場合は特別に降車時刻を指定する．この利用者 u_i の降車時刻に対する $delay_i$ を小さくするために重み係数 γ_i を極端に大きくする．これは，利用者が列車に乗り遅れる可能性をなくすためである．

このように，乗降車要求の違いを利用者の属性として表すことにより，β, γ, δ の値を変化させ，適切な運行計画を作成することができる．

2.3 運行計画

運行計画は，各バスの運行経路と利用者の希望乗降車点へのバス到着予定時刻からなる．利用者が乗降する各停留所間は最短経路を走行するものとする．全ての停留所間の最短経路の移動時間は前処理で求めておく．すなわち，走行経路とバスの出発時刻が与えられれば，各利用者の希望乗降車点へのバスの到着時間が分かり，運行計画全体が決定される．

しかし，最短経路で運行を続けることによって，特定の利用者 u_i の $early_i$ の値が著しく大きくなる場合がある．つまり，利用者 u_i の希望乗車時刻より，かなり早めにバスが来てしまうことがある．この解決策として，バスを適切な場所で待機させ $early_i$ を小さくする方法も考えられるが，乗車中の利用者 u_j があった場合，$slack_j$ が大きくなる．バスが待機するかどうかは，乗客サービスの観点からシステム（又はオペレータ）が決定している．

3 経路探索の実際的計算手法

利用者の乗降車要求が与えられた時に，運行コストを最小にする運行計画を構築する問題は NP 困難である．本問題において，全ての運行経路を調べて最適解を見つけようとすると，m 人の利用客に対して $(2m)!/2^m$ 通り存在し，計算時間が極めて大きくなる．本問題は高速な処理が求められるため，これは現実的ではない．ところが，実用面では最適解ではなく近似解で十分な場合がほとんどである．

近年，メタヒューリスティックスと呼ばれる発見的な計算手法が提案されている．これには，遺伝的アルゴリズム (GA)，シミュレーテッド・アニーリング (SA)，局所探索（山登り法）などがある．本デマンドエンジンでは，そのひとつであるタブー探索法 (TS) を用いる．本システムのように，利用者の属性に応じてきめの細かい運行計画を作成するには，アルゴリズム自体が単純な方が実際的なためである．

タブー探索法は局所探索を基本としている．局所探索法は，最初に初期解を構築してそれを暫定解とし，解空間内で暫定解の近傍を探し，よりよい解を採用するということを繰り返しながら最適解を求めようとする方法である．しかし，この方法は局所最適解（すなわち近傍に改善解が存在しない解）に到達すると，そこから抜け出せないという欠点を持つ．

タブー探索法は，一度訪れた解を記憶し，一定の間再び訪れることを禁止する．暫定解の更新時においては，解の悪化も許容する．これにより，局所最適解を乗り越えて探索を続けることが可能であり，局所探索法の欠点を克服している．

この節では，まず，初期解の構築方法を述べ，次に，解の近傍を定義する．さらに，タブー探索の詳細と，暫定解の更新方法について述べる．

3.1 初期解の構築

ここでは簡単のためにバスが 1 台の場合について述べる．利用者 $u_i (1 \leq i \leq m)$ の乗車点を p_i，降車点を d_i とし，バスの出発点，終着点をそれぞれ S, F とする．以下ではバスの運行経路を $p_i, d_i (1 \leq i \leq m)$ の順列で表わす．

初期解の構築には挿入法と呼ばれる単純な方法を用いる．まず，利用者 (u_1) の乗降車点 p_1, d_1 を S と F の間に挿入し，利用者数 1 人の解 S_1 を構築する．次に，利用者 u_2 の乗車点 p_2 を解 S_1 に対して，運行コストが最小とな

る位置に挿入し, 続いて降車点 d_2 を乗車点 p_2 の後方において運行コストが最小となる位置に挿入し, 解 S_2 を構築する. 利用者 $(u_3, ..., u_m)$ について乗降車点の挿入を繰り返し解 S_m を構築する. この S_m を初期解とする.

バスが複数台の時も上と同様にして初期解を構成するが, 初期解の実行可能性は次の3点で保証する. 同一利用者の乗降車点が, 同一バスの運行経路上に存在すること. 同一利用者の降車点は, その乗車点の後方に存在すること. 同一バスの乗車人数が, 定員を超えないことである.

3.2 近傍の定義と暫定解の更新

ここでは, 暫定解の近傍を定義する. まず, 現在の解 S_t から一人の乗降車点 p_i, d_i を取り除き $m-1$ 人の解 S'_t とする. 次に, 運行コストが最小になるように S'_t に乗車点 p_i を挿入し直し, p_i の後方に, 運行コストが最小となる降車位置を探し d_i を挿入する. これが, ひとつの近傍解である. 次に解 S_t に戻り, 別の利用者の乗降車点を取り除き上と同様にして近傍解を構築する. このようにして, m 個の近傍解を得る (図 1 参照).

S' に対して, 一人の乗降車点 p_i, d_i を挿入可能な位置は各々$O(m)$ 箇所存在するから, 全ての場合を調べると $O(m^2)$ の手間がかかり, m 人に対して繰り返すと近傍解全てを求める手間は $O(m^3)$ となる. しかし, 中規模以上の問題例への適用を考えて, 上では, 最初にコスト的に最適な場所を探して乗車点 p_i のみ挿入し, 乗車点を固定した上で降車点 d_i の挿入場所を探すことでこの手間を $O(m^2)$ に減らしている.

利用者 m 人の暫定解に対して m 個の近傍解を構築する方法は上で述べた. この近傍解の中で, 運行コスト最小のものを新たな暫定解として更新する. (新しい暫定解は, 直前の暫定解よりも運行コストが大きくなることがある. これは, タブー探索の特徴のひとつである.) 暫定解の更新を一定回数繰り返し, 発見した暫定解の中で運行コスト最小なものを運行計画として出力する.

3.3 タブー探索法

タブー探索法では, 過去に訪れた解を記憶し一定期間再び訪れないようにしている. これは, 訪れたばかりの解を近傍解から取り除くことで実現できる. このためにタブーリストと呼ばれるものを用いる.

ここで用いるタブーリストは次のようなものである. タブー探索における現在の暫定解を S_t とする.S_t は直前の暫定解 S_{t-1} に対し, ある一人の利用者 u_i の乗車点 p_i と降車点 d_i を挿入し直したものとなっている. このとき暫定解 S_t における p_i の直前の乗降車点 a と d_i の直前の乗降車点 b をタブーリストの中に覚えておき, p_i の直前の乗降車点が a かつ d_i の直前の乗降車点が b であるような解を近傍の中から排除する. (図 1 では,$a = p_1, b = d_2$ である.)

タブーリストは2次元の配列 $T[a, b]$ で表現する. $T[p_i, a]$ と $T[d_i, b]$ に整数値 T_s (タブースパン) を書き込む. また,$T[p_i, a]$ と $T[d_i, b]$ 以外で正の値を持つ T の要素は, 暫定解を更新する毎1ずつ減じる. つまり, タブーリストは, 利用者 u_i の乗降車点 (p_i, d_i) が過去 T_s 回の暫定解の更新時にどの乗降車点の直後に挿入し直されたかという情報を保持することになる.

近傍解を構築する時には, 利用者 u_i について,$T[p_i, a] > 0$ かつ $T[d_i, b] > 0$ であるような a と b の直後にそれぞれ p_i, d_i を挿入しないようにする. このように, 挿入し直した乗降車点の情報を保持することで, 過去に訪れた解を一定期間再び訪れないようにしている.

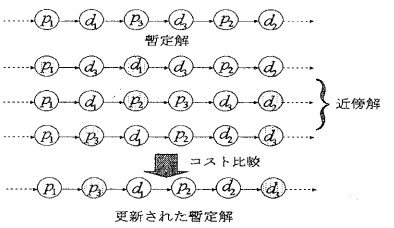

図 1: 暫定解の更新

4 計算機実験

4.1 計算時間と解の近似比の評価

本節では全解探索 (分枝限定法) により最適解を求め, デマンドエンジンを用いて得られた解の運行コストと比較する. また, その時の計算時間についても比較を行う.

実験は, 利用者数ごとに乗降車要求を１００通りランダムに発生させ, 分枝限定法, デマンドエンジンそれぞれに運行計画を作成させた. タブー探索における暫定解の更新は２００回である.

また, デマンドエンジンで得られた運行計画の運行コストと, 全解探索で求めた最適解の運行コストの比を調べた (表 1) . 実験結果を見ると, デマンドエンジンがほぼ最適な運行計画を与えていることが分かる.

計算時間については, 平均時間と最悪時間を表示した. この結果から, 利用者数の増加に伴い計算時間に大きな差がみられ, デマンドエンジンの実用性が分かる. (利用者数が２０人を超えると全解探索は計算時間が膨大になり実際上計算不可能であった.)

表 1: 計算時間比較（秒）及び近似比率

利用者	計算時間				運行コストの近似比	
	提案法		全解探索			
	平均	最悪	平均	最悪	平均	最悪
7	0.136	0.15	1.2	17.11	1.003	1.228
8	0.204	0.22	33.6	1500	1.007	1.173
9	0.276	0.30	378	4401	1.010	1.641
10	0.378	0.40	4147	20478	1.032	1.342
20	2.79	2.93				
30	9.01	9.58				
50	34.9	38.4				

4.2 タブー探索の効用

タブー探索法は, 局所最適解に陥ってしまうという局所探索法の欠点を克服していることは先に述べた. その効果を, タブー探索を用いた場合と局所探索のみの探索の場合とで, 作成される運行計画の運行コストを比較することで示す. ($T_s = 0$ とすることで, 本デマンドエンジンは局所探索と同じ動作をすることに注意されたい.)

実験に用いた乗降車要求は, 中村市で実際に発生したものを用いている. 全て通常の乗降車要求であり, 各種係数は次のとおりである. 重み係数は, バスの走行時間より利用者を優先する立場から $(\alpha, \beta, \gamma, \delta) = (1.0, 1.5, 4.0, 4.0)$ としている. また,$T_s = 0, 15, 25$ とし 50 回暫定解を更新して暫定解全ての運行コストを表示した（図 2）. 図２では, 横軸が暫定解の更新回数, 縦軸が運行コストを示している.

この実験により, タブー探索を行った結果, 局所最適解から一旦運行コストが悪化する暫定解をみつけ, やがて局所最適解より運行コストの小さな解を見つけている様子がわかる. これは実験を行った乗降車要求のほぼ全てに対して見られる傾向である.

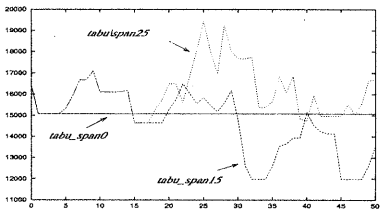

図 2: タブー探索の効果 ($T_s = 0, 15, 25$)

4.3 タブースパンの特性

この節では, タブースパンを変化させ, 最適解の発見回数を調べた. 利用客数は, 中村市の一回のバス運行時の平均利用者数６人と固定し, ３０種類の乗降車要求に対して各タブースパンごとに実験し, その中で最適解の得られた回数を調べた（図 3）タブースパンが短い時はタブー探索の効果があまり表れていない. また, 本システムはタブーリストに暫定解を構築する際の乗降車点の挿入場所の情報を蓄える. その結果, 実際に訪れた解とその周辺にある運行コストの低い解も解空間から除いている. その結果, タブースパンが大きすぎると運行コストの小さな多くの解を制限してしまうことになり, 最適解の発見の障害になることが分かる（図２）. この実験から, タブースパンを１２程度とするのが適当と考えられる.

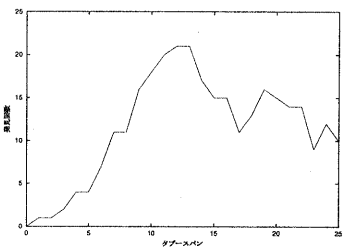

図 3: タブースパンと解の発見回数

4.4 バス２台の運行計画

中村市で運行しているデマンドバスでは, 現在のところバスは１台である. 今後, 利用者の増加や, 今回の対象地域より規模の大きな地域での適用を考えると, バス１台では利用者の乗降車要求に十分応えることが出来ない可能性がある. 短時間に乗降車要求が集中すると *slack* と *delay* の値が極端に大きくなり, 実用的ではなくなるからである. この解決法としてバスの運行台数を増やす

表 2: バス２台運行時の *delay* の値 （分）

利用者	*delay*			
	１台		２台	
	平均	最悪	平均	最悪
10	4.22	80.8	1.25	20.7
15	7.70	85.3	1.76	30.5
20	9.55	79.5	2.63	41.4
25	13.0	99.0	3.17	47.6

表 3: バス２台運行時の *slack* の値（分）

利用者	*slack*			
	１台		２台	
	平均	最悪	平均	最悪
10	10.6	88.6	3.10	31.5
15	11.2	94.3	4.76	59.7
20	13.4	98.3	7.09	59.3
25	13.7	90.4	8.94	73.0

ことが考えられる．そこで，今回のシステムで利用者数を増やし，バス１台の場合と２台に増やした場合でエンジン性能を比較した．

実験では，利用者数ごとに，一時間の間に乗降車要求を３０通りずつランダムに発生させた．（通常の乗降車要求のみで，オンライン乗降車要求，特急利用者は含んでいない．）この時，*slack* と *delay* の全利用者の平均値を調べ，バス台数を増やした効果を比較している．

実験の結果 *slack* と *delay* の平均値は大幅に軽減されていることが分かる．また，*delay* の平均値が *slack* の平均値より大幅に小さいのは $(\beta, \gamma) = (1.5, 4.0)$ であるため *delay* を小さくする運行計画がつくられた為と考えられる．一方で，*slack* と *delay* 最悪値は，その平均値より極めて大きな値となる問題を抱えている．

原因としては，利用者の属性に対して重み係数は変化するのに対して，*slack* と *delay* の値に対して変化していないことがあげられる．例えば，*delay* が５分の利用者６人と *delay* が３０分の利用者１人は運行コストを計算する際に同等である．つまり，特定の利用者の *slack* と *delay* の値のみが大きくなってしまう傾向を持つということである．この対策については現在，実験・考察を進めている．

5 まとめ

中山間地におけるデマンドバスシステム導入において，提案法によって高速に，ほぼ最適な運行計画が得られる事を示した．本システムで用いたタブー探索法の有効性を示した．現在は，１台のバスでカバーできる範囲で運行しているが，利用者の増加，他地域での実用を考え合わせ，バス２台に増やした実験を行い，現エンジンの問題点を抽出した．これらの問題点を考慮した，新しいデマンドエンジンの開発が今後の課題である．

謝辞 本実験にあたって，ご協力いただいた高知県企画振興部情報企画化および土木部道路課，中村市，高知西南交通株式会社の関係各位に深く感謝いたします．また，本研究の初期段階は名古屋大学大学院工学研究科片山恭介氏の修士学位論文によるところが大きい．記して感謝いたします．

参考文献

[1] M. Charikar and B. Raghavachari, "The finite capacity dial-a-ride problem," Proc. 37th FOCS, pp. 458-467, 1998.

[2] M. W .P Savelsbergh and M. Sol, "The general pickup and delivery problem," Transportation Science, 29(1), pp. 17-29, 1995.

[3] P. Healy, R. Moll, " A new extension of local search applied to the dial-a-ride problem," European Journal of Operation Research, 83, pp. 83-104, 1995.

[4] C.R. Reeves 編, 横山隆一, 奈良宏一, 佐藤晴夫, 鈴木昭男, 萩本和彦, 陳洛南 訳, "モダンヒューリスティックス一組合せ最適化の先端手法" 日刊工業新聞社, 1997.

[5] 片山恭介, "Dial-a-Ride システムにおける経路探索に関する研究," 名古屋大学大学院工学研究科修士学位論文, 2000.

[6] 誉田安秀, "中山間地におけるデマンドバスシステムの開発および実証実験," Matsushita technical Journal Vol.46, No.4 pp. 105-113, 2000.

[7] 室田一雄 編, "離散構造とアルゴリズム," 近代科学社, pp. 171-230, 1995.

[8] 柳浦睦憲, 茨木俊英, "組合せ最適化問題に対するメタ戦略について" 信学論 Vol. J83-D-I No.1, pp. 3-25, January 2000.