モバイルコンピューティングと
ワイヤレス通信　23−32
高度交通システム　11−32
（2002．11．29）

デマンドバスシステムのための最適経路選択法の実装とそのオンライン配車システムへの応用

中谷昭彦 †　藤田 聡 †

† 広島大学大学院 工学研究科 情報工学専攻
〒 739-8527 広島県東広島市鏡山一丁目 4-1
{nakatani,fujita}@se.hiroshima-u.ac.jp

あらまし: 本稿ではデマンドバスの運行管理問題について考察する。以下ではまず、1台のバスに対する運行距離最小化問題を厳密に解くための最適化解法を提案する。提案手法は動的計画法と A* アルゴリズムに基づいており、50 人分の乗車要求に対する最適解を1秒以内で出力することができる。本稿では次に、実装された解法器を利用したデマンドバス配車システムの提案をおこなう。本システムで用いられる発見的な配車手法の性能は、シミュレーションにより実験的に評価される。

An Exact Algorithm for Minimum-Cost Routing Problem and its Application to Online Allocation of Demand-Buses

Akihiko Nakatani† Satoshi Fujita†

† **Graduate School of Engineering, Hiroshima University**
Kagamiyama 1 chome 4-1, Higashi-Hiroshima, 739-8527 Japan

Abstract: In this paper, we propose an exact algorithm for solving the minimum-cost routing of a single bus with designated deadlines. The proposed method is based on the dynamic programming and A*, and can output an optimal solution for 50 passengers in one second. The method is applied to an online bus allocation system, that is used in demand-bus management systems as a basic component, and the performance of the overall system is experimentally evaluated by simulation.

1 はじめに

騒音問題や大気汚染などへの対策の一環として、市街区域への自家用車の乗入れ規制が国内外の多くの都市で実施・検討されている[1]。このような規制の導入によって、CO_2 排出量が大幅に抑制されるなど環境に対する悪影響が軽減することが期待されるが、その一方で公共交通機関への依存度が相対的に増すことから、その導入には交通インフラの整備とセットにした施策の検討が不可欠であると考えられる。郊外の団地と市街地とを結ぶ新しいインフラのあり方として注目されている方式のひとつにパーク・アンド・ライド (P&R) 方式がある。P&R 方式では、各利用者は自家用車で直接市街地に乗り入れる代わりに、市街区域外に用意された (無料) 駐車場に車を停め、そこから市街区域内までは軌道車両や定期路線バスなどを利用して移動するという方法がとられる[2]。しかしこの方式では、市街区域内での移動に関する自家用車の利便性が十分補償されているとはいえず、そのような移動をサポートするためには、今後さらなるインフラの整備が必要であると考えられる。本稿では、利用者の市街区域内での移動を効率的にサポートするための具体案を提示し、その実効性や実現可能性に関する予備的な評価をおこなう。本研究の最終目的は、自家用車の乗り入れが制限された区域内で発生する利用者からの動的な輸送要求に対し

[1] たとえばフライブルク市などの事例が有名である。詳細については http://www.midi.co.jp/ akuaku/teigen.html あるいは http://www.pal.or.jp/env/report2.html を参照のこと。

[2] この種のサービスは、たとえば金沢市では平成 8 年からすでに実施されている。

て、高い信頼性と高い即応性とをあわせもった新しい公共交通システムを構築していくことである。

本研究では、市街区域での乗客の輸送にデマンドバスシステムを適用することを考えている。具体的には、デマンドバスの運行管理問題 (以下、DSP) に対する効率的な解法を与えることによって、利用者の利便性を確保しつつ運行効率を向上させるというアプローチをとる。ここで DSP とは、特定の地域内から出された乗降者要求に対して乗合型の車両の割当てをおこない、各車両の巡回経路を適切に決定・制御する問題である (文献では Dial-a-Ride 問題と呼ばれることもある)。DSP に関してはこれまでに様々な研究がなされているが、乗客の到着モデルや用意される車両数などによって、アルゴリズムの設計方針は大きくかわってくる。１台の車両の経路をオフライン的に求めるための手法としては、動的計画法 (DP) による最適解法 [7, 8] や分枝限定解法 [4] のほか、遺伝的アルゴリズム (GA) を用いた発見的解法 [5] などが提案されている。また車両が複数ある場合の配車制御を含めた研究に関しては、集合分割法を用いた解法 [1, 3]、GA による解法 [10]、エージェントを利用した解法 [6] などが提案されている。

本稿で提案する手法は、DP を用いた厳密解法に基づいている。手法の概要は以下の通りである：1) 利用者からの要求はすべてコントロールセンター (CC) に集められ、CC は適切な戦略にしたがって各要求を車両に割り当てる。2) 要求を受け取った各車両は、与えられた要求たちからなるインスタンスを DP を用いて厳密に解く。3) ただし制約条件が満たされない場合、その車両は与えられた要求を CC に返却し、要求を再び受け取った CC は、制約を満足する割当が可能な車両が見つかるまで同様の処理を繰り返す。本稿ではまず最初に DP を用いた１台の車両に関する厳密解法を PC 上に実装する。実験の結果、割り当てられる要求の数が 50 程度であれば、1 秒程度で最適解が求められることがわかった。次にその結果に基づいてシステム全体のシミュレーションをおこない、いくつかの発見的割当て手法の性能について実験的に評価する。

本稿の構成は以下の通りである。2 節ではモデルについて説明する。3 節では 1 台のバスに対する厳密解法の提案をおこなう。4 節では配車システムの概要を述べ、そこで用いられる発見的割当て手法の性能は 5 節で評価される。最後に 6 節でまとめと今後の課題を述べる。

2 モデル

2.1 距離グラフ

デマンドバスが運行される地域の地図から、以下のような手順で有向グラフを構成する: まず最初に、地図中に乗車ポイントと降車ポイントをそれぞれ複数設定する[3]。次にバスが通行可能な道路を地図からすべてピックアップし、選択された道路の分岐点にあたるすべての場所に分岐ポイントを設定する。設定されたすべてのポイントを頂点集合としてもち、それらのポイントを両端点とするすべての道路のセグメントを有向枝集合としてもつ多重有向グラフを考え、これを $D = (V(D), E(D))$ であらわす。本稿では D の強連結性を仮定する。グラフ D の各枝には、その枝に対応する道路セグメントを通過するのに要する平均時間がコストとして与えられる。有向枝 e のコストを $w(e)$ であらわす。以下では、D 上の乗車ポイントの集合を U^+、降車ポイントの集合を U^- であらわす ($U^+ \cup U^- \subseteq V(D)$ であることに注意)。トが含まれているものとし、

$V(D)$ 中の任意の 2 頂点 u, v に対して、u から v へ移動するための (コスト関数 w の意味での) 最短経路を考え、その長さを $d(u, v)$ と記す。有向グラフ D に対する**距離グラフ** G とは、頂点集合が $U^+ \cup U^-$ であるような重み付き有向完全グラフであり、u から v へ向かう有向枝の重みが $d(u, v)$ となるようなもののことをいう。本稿では、デマンドバスのスケジューリングはすべて距離グラフ G 上でおこなわれるものとする。すなわち、オンライン的に入力される要求に対するスケジュールの見直しは、バスが乗車ポイントか降車ポイント上に存在するときにのみおこなわれ、ポイント間の移動中にはスケジュールの変更はいっさいおこなわない。また 2 ポイント間の最適経路選択は、(たとえばオンライン的に獲得可能な渋滞情報などをもとにした) 適切な下位層のナビゲーションメカニズムによって実現されるものと仮定する。

2.2 利用者

利用者の可算無限集合を P であらわす。利用者 p ($\in P$) は、頂点 $v_s(p)$ ($\in U^+$) にオンライン的に到着し、適当な待ち時間の後、コントロールセンター (CC) によって割り当てられたバスに乗車し、目的頂

[3] あるポイントが乗車ポイントと降車ポイントを兼ねてもよい。また道路をはさんで上流方向と下流方向に別々のポイントを設定してもかまわない。

点 $v_d(p)$ ($\in U^-$) に移動する。p の頂点 $v_s(p)$ への到着時刻を $t_s(p)$ であらわす。利用者 p は、乗車ポイント $v_s(p)$ への到着時に、目的地に到着するまでのデッドライン $DL(p)$ ($> t_s(p)$) を指定することができる。スケジューラは、利用者 p を時刻 $DL(p)$ までに目的頂点 $v_d(p)$ に届けるようなスケジュールを生成するか、デッドラインの達成が不可能ならばそのことを利用者に対して通知しなくてはならない (スケジューラの詳細については後述)。利用者 p に関する情報 $(v_s(p), v_d(p), DL(p))$ は、利用者の頂点 $v_s(p)$ への到着後直ちに各ポイントに設置された端末から CC に送られる。CC では、送られてきた情報をもとにスケジュールの調整をおこない、(もし必要であれば) 各バスに対してスケジュール変更等の適切な指示を与える。以下では、情報伝達やスケジューリング等にかかる時間はバスの移動時間に比べて十分短いものと仮定し、CC からバスへの指示は時刻 $t_s(p)$ に与えられるものとする[4]。ただし前述のように、CC からの指示がバスのスケジュールに反映されるのは、バスの (時刻 $t_s(p)$ の時点での) 目標頂点への到着後になる。

2.3 バス

バスの集合を $\mathcal{B} = \{b_1, b_2, \ldots, b_m\}$ とする。各バスにはそれぞれ定員が定められており、与えられた定員をこえての乗車は一切できないものとする。バス b_i の定員を $c(b_i)$ と記す。乗車ポイントで待っている利用者 p は、バス $b \in \mathcal{B}$ がそのポイントに停車した際に、降車があればそれに引き続いて、総待ち時間の長い利用者から順に定員に達するまで乗車することができる[5]。

距離グラフ G 上でバス b が訪問した頂点の系列を $P = \langle u_0, u_1, \ldots, u_x \rangle$ とするとき、その総移動距離は

$$Z(P) \;=\; \sum_{i=1}^{x} d(u_{i-1}, u_i)$$

で定義される。以下では、バスが距離 1 だけ進むのに要する時間を 1 単位時間とし、各ポイントでの乗降車にかかる時間は乗降人数によらず一定値 Δ とする。したがってバス b が頂点系列 P に沿って移動するのに要する総経過時間は、$Z(P) + x\Delta$ であらわすことができる。本稿で対象とする最適化問題 DSP は、各車両 b_i の総移動距離を Z_i としたとき、その和 $\sum_{i=1}^{m} Z_i$ を最小化する問題である。ただし制約条件として各乗客の設定したデッドライン制約をすべて満たす必要があり、さらに乗客の到着はオンライン的におこなわれ、スケジューラはすべての要求の系列を前もって知ることはできないものとする。

[4] 具体的な時間のオーダーとしては、頂点間のバスの移動には数分程度、情報伝達やスケジューリングには数秒程度の時間がかかるものと考えている。

[5] バスの現在の行き先方面ごとに乗車のコントロールをおこないたい場合には、行き先方面ごとに別の乗車ポートを用意するなどの方法で対応することができる。

3 最適化アルゴリズム

3.1 動的計画法の概要

動的計画法 (DP) を用いた 1 台の車両に対するオフライン DSP の厳密解法が Psaraftis[8] と Desrosiers ら [2] によって提案されている。ここで “オフライン DSP” とは、すべての乗客の乗降車要求が求解時にあらかじめわかっているという仮定のもとで解かれる DSP のことである (通常の DSP はオンライン DSP である)。我々が想定しているモデルでは乗降車ポイントにいる間に再スケジューリングがおこなわれるため、新たなポイントに到着するごとに、現在までに出されていてまだ叶えられていない要求たちに関するオフライン DSP を解くことで、デマンドバスの効率的な運行制御をおこなうことができると考えられる (ただし将来の入力は予測できないため、現在の入力に対する最適な選択が将来を含めた入力系列全体に対しても最適であるかどうかは保証されない)。

DP を用いた解法のアイデアは、初期状態から到達可能な各状態に対して、その状態に至るまでの G 上の移動距離が最短であるような遷移系列をひとつだけ保持することによって、考慮すべき組合せ数を大幅に削減することにある。ここで初期状態とは、移動距離がゼロであるような状態のことであり、最終状態とは、与えられた要求をすべて達成した (すなわち、すべての利用者をその乗車ポイントから降車ポイントまで運び終わった) 状態のことである。DP を用いることによって探索すべき組合せの数は大幅に減少するが、可能な状態数自体は、依然として処理すべき乗客数に関して指数関数的に増加する。さらに、過去に提案された各解法では状態の探索は幅優先探索的になされており、最適解に至ることのない不要な状態が数多く生成されてしまうという問題点もある。このような問題点を解決するため本稿では、DP による探索に A* アルゴリズムを導入するという

アプローチをとる (A* の詳細については後述する)。

3.2 状態の定義

A* アルゴリズムの説明をおこなう前に、DP で用いられる “状態” の定義をしておく。スケジューリングをおこなう時点で与えられている乗客数を n とする。ある時点におけるバスの状態 S は、各乗客の状態をあらわす n 個の変数 $s_1, s_2, \ldots, s_n$ と、その時点でバスが停車している G 上の頂点 ($\in U^+ \cup U^-$) とから構成される $(n+1)$ 項組である。ここで s_p $(1 \leq p \leq n)$ は乗客 p の状態をあらわす 3 値の変数であり、乗客 p がまだバスに乗車していないとき 0、乗車中のとき 1、目的ポイントに到着しているとき 2 をそれぞれ値としてとる。なお各乗客の状態は、現在停車中のポイントにおける必要な乗降車をすべて終え、そのポイントを出発する直前の状態であるとする。

DP による解法では、各状態 S にラベルの有限集合 $\mathcal{L}(S) = \{l_1, l_2, \ldots, l_x\}$ が割り当てられる。$\mathcal{L}(S)$ 中の各ラベル l は、初期状態から状態 S に至るひとつの状態遷移系列に対応しており、以下の 3 つの要素から構成される：1) その系列上での S の先行状態 S'、2) その系列にそってバスが移動したときの総移動距離 $Z(l)$、および、3) その系列上で訪問したのべ頂点数 $X(l)$。ただし、ある状態遷移系列に対応するラベル l が集合 $\mathcal{L}(S)$ に含まれるのは、以下のような条件が満たされている場合に限られる：

- $\mathcal{L}(S)$ 中に、$X(l') = X(l)$ かつ $Z(l') \leq Z(l)$ となるような状態遷移系列 l' が含まれていないこと。したがって、訪問したのべ頂点数が同じ遷移系列については、Z の値が最小のものがひとつだけ $\mathcal{L}(S)$ に含まれることになる。さらに n 人の乗客の要求をすべて満たすためには高々$2n$ 個の頂点を訪問すればよいから、$\mathcal{L}(S)$ の要素数は任意の S に対して $2n$ 以下であることもわかる。

- l で示される状態遷移系列にそってバスが移動したときに、デッドライン制約が満足できなくなるような乗客が存在しないこと。すなわち、状態 S で目的頂点に到達していない乗客 p の中に $DL(p) \leq Z(l) + X(l)\Delta$ であるようなものが存在するならば、ラベル l は $\mathcal{L}(S)$ には含まれない。

3.3 A* アルゴリズムの概略

あるグラフ上に設定されたスタート節点からゴール節点までの最短経路を効率よく発見する問題を考える。スタート節点から節点 x までの最短経路の長さを $g(x)$ と記す。ダイクストラ法による探索では、$g(x)$ の値の小さいものから順に展開していくことでゴール節点までの最短経路を計算していくが、もし各節点 x からゴール節点までの距離 $h(x)$ が何らかの形で予測できるならば、その予測値 $\hat{h}(x)$ を用いることで、探索をより効率的におこなうことができる。このような予測に基づく探索のことを一般に発見的探索と呼ぶ。本稿で着目する A* アルゴリズムは、人工知能などの分野で幅広く用いられている典型的な発見的探索手法のひとつである。

A* アルゴリズムでは、スタート節点から節点 x までの距離の予測値 $\hat{g}(x)$ と、節点 x からゴール節点までの距離の予測値 $\hat{h}(x)$ とを利用することにより、最適解に関連する見込みのない無駄な探索を避けながら、最短経路を効率よく求めることができる (ここで $\hat{g}(x)$ はそれまでに発見された経路の中で最短なものの長さであり、実際の x までの距離はそれよりも小さいかもしれないことに注意されたい)。具体的には、節点 x を通る最短経路の長さは $\hat{g}(x) + \hat{h}(x)$ によって “推定” されることになる。以下では、推定された最短経路の長さを $\hat{f}(x)$ と記すことにする。節点の展開は $\hat{f}$ の値の小さい節点から順におこなう。いま節点 x が展開されるとしよう。具体的には x の展開は以下のような手順でおこなわれる：まず節点 x のすべての後続節点を調べ、その中にはじめて見つかった節点 x_i があれば、$\hat{f}(x_i)$ を計算して候補リストに加える。過去に到達したことのある節点 x_j については、枝 (x, x_j) を通る最短経路長の推定値 $\hat{f}(x, x_j)$ とこれまでに得られた x_j を通る最短経路長の推定値 $\hat{f}(x_j)$ とを比較し、前者が小さければ候補リストに加える。なお、すべての後続節点に関する処理が終了した後で節点 x は候補リストから到達済みリストに移されるが、一度到達済みリストに加えられた節点であっても、その節点に至る “より短い経路” が見つかれば、その節点は新たに候補リストに入れなおさなくてはならないことに注意されたい。

上記アルゴリズムにおいて、すべての節点 x において $\hat{h}(x) \leq h(x)$ が成立するとき、A* アルゴリズムと呼ばれる (そうでないときは A アルゴリズムと呼ばれ、最適性の保障はできなくなる)。一般に $\hat{h}(x)$ の精

度が悪い場合には探索の枝刈り効果は小さく、$\hat{h}(x)$ が $h(x)$ に近いほど探索効率は高まる。ただし $\hat{h}(x)$ が $h(x)$ を超えるような値を与える場合には誤った情報を与えたことになり、探索効率が悪くなったり、最適解が見つからないこともある。

3.4 提案手法

DSP の DP 解法に A* アルゴリズムを適用する方法は以下の通りである。まずスタート節点 (初期状態) からの予測距離をあらわす関数 $\hat{g}$ については、その状態に至るまでに通ってきた G 上の総移動距離がそのまま使用できる。一方、状態 S からゴール節点 (最終状態) までの予測距離をあらわす関数 $\hat{f}(S)$ は以下のように計算すればよい：まず状態 S において停車している G 上の頂点を v_s とし、状態 S から最終状態に至るまでに停車しなければならないポイントの集合を $U(S)$ ($\subseteq V$) とする。$U(S)$ の中には、1) その時点で乗車していない乗客たちの乗車ポイント、2) その時点で乗車していない乗客たちの降車ポイント、および 3) その時点で乗車中の乗客たちの降車ポイントの 3 種類のポイントが含まれている。頂点集合 $U(S) \cup \{v_s\}$ によって誘導される G の誘導部分グラフを G' とし、G' 上の最小生成木のサイズ (木に含まれる枝重みの総和) を $\phi(G')$ とする。このとき $\phi(G')$ は、v_s を出発して $U(S)$ のすべての頂点を訪問し終わるまでに移動しなくてはならない距離の下限を与えている。したがって、

$$\phi(G') \ \leq \ h(S)$$

より、$\phi(G')$ を $\hat{h}(S)$ として使用することができる。

関数 $\phi(G')$ のゴールまでの予測距離としてのよさは、バスの定員が十分大きい場合は、以下の式によって理論的に保証することができる：

$$h(S) \ \leq \ 4\phi(G') \qquad (1)$$

上式が成立する理由は以下の通りである。まず G' 上の最短のハミルトン閉路の長さは $2\phi(G')$ 以下である (得られた生成木上を根から順に深さ優先探索して根に戻るまでに、木の各枝は高々2 回しか通らない)。次にバスが G' 上の任意のハミルトン閉路を 2 回巡回すれば、すべての乗客は乗車ポイントから降車ポイントまで移動することができる。したがって v_s を出発して G' 上の頂点をすべての要求を満足するように巡回する最短の道の長さは、高々 $4\phi(G')$ である。

表 1: 厳密解法の性能 (グラフ G_2 の $n = 50$ については、メモリ不足のため実行できず).

	グラフ G_1		グラフ G_2	
n	時間 [sec]	状態数	時間 [sec]	状態数
1	0.02	2	0.01	2
2	0.00	3	0.00	3
3	0.00	8	0.00	7
4	0.00	15	0.00	12
5	0.00	38	0.00	21
6	0.00	150	0.00	31
7	0.01	469	0.00	183
8	0.02	999	0.01	519
9	0.10	4032	0.05	1869
10	0.16	5751	0.29	8533
15	0.16	5199	2.19	45285
20	0.37	9951	1.93	35160
25	0.37	9347	10.03	145521
30	0.44	10266	11.27	154454
35	0.51	10203	16.56	200871
40	0.52	9600	21.51	239003
45	0.58	9858	59.19	594698
50	0.60	9700		

3.5 解法の性能

提案する厳密解法の性能をその実行時間により評価した。実験に用いた PC の仕様は、CPU が Pentium4 2.26GHz、メモリが 1024Mbyte である。実験では、ランダムに生成された頂点数 10 の距離グラフ G_1 とランダムに生成された頂点数 14 の距離グラフ G_2 のそれぞれに対して、乗客からの要求をひとつずつ加えていくことによって得られるそれぞれの n に関するインスタンスに対してアルゴリズムを走らせ、その実行時間と展開された状態数とを調べた。ここでバスの定員は 100 名とし、各乗客の設定するデッドラインはそれぞれ無限大とした[6]。結果を表 1 に示す。

この表から、アルゴリズムの実行時間はインスタンスによって多少変わってくるものの、乗客数が 20 名程度であれば数秒以内に最適解を求められること

[6] デッドラインを有限値に設定することで A* による枝刈りは働きやすくなるため、デッドラインを無限大にしたほうが、展開される状態数と実行時間は一般に大きくなる傾向にある。ただし、デッドライン制約が厳しすぎてすべてのデッドラインを満足する解が存在しない場合などには、DP アルゴリズムはすべての可能性を探る方向に動作してしまうため、実行時間が逆に長くなる場合もある。

がわかる。また頂点数が 10 の場合は、乗客数が 50 名になっても 1 秒以内で解が求められており、提案手法の効果が十分あらわれていることがわかる ($n = 40$ のときに展開される状態数は、頂点数 14 のインスタンスの数%程度に抑えられている)。

4 配車システムへの応用

4.1 配車システム

本稿で提案するデマンドバスシステムのための配車システムは、コントロールセンター (CC) に置かれたディスパッチャと各車両ごとに用意された車両ルータとから構成される。車両ルータは、ディスパッチャから割当てられた乗降車要求たちに対する最適経路を、前節の手法を用いて厳密に計算することができる。乗客からオンライン的に出される要求に対する処理フローは以下の通りである：

1. まず乗客からの要求は全て CC へと集められる。CC 上のディスパッチャは、受け取った要求の中身と各車両に関する情報 (位置、乗車人数、定員) とをもとに、適切な戦略にしたがって要求を各車両にオンライン的に割り当てる (配車のための具体的な戦略については 4.2 節で詳しく述べる)。

2. 各車両ルータは、ディスパッチャから割り当てられた要求たちからなるインスタンスを前節の解法を用いて厳密に解く。

3. 与えられたインスタンスに実行可能解 (すなわちすべてのデッドライン制約を満足する解) が存在しない場合、車両ルータは与えられた要求をディスパッチャへと返す。そうでなければ、車両ルータは得られた最適解をその車両の新たな運行経路として設定しなおす (なおこの設定が実際にスケジューリングに反映されるのは、前述の通り、次の目的頂点への到着後となる)。

4. 車両ルータから要求を返されたディスパッチャは、実行可能解を出力する車両ルータが見つかるまで、同様の処理を繰り返す (ただし実際には、試行回数に上限を設けておき、固定された回数だけ試行を繰り返すことになる)。

4.2 配車方法

本稿ではディスパッチャで用いられる具体的な配車方法として、以下の 5 種類の配車方法に着目する：

Weight: 利用者から受け取った要求を乗客数が最小の車両へと割り当てる方法。

Random: 乗客数や車両位置等によらずランダムに割り当てる方法。

GetOn: G 上の頂点集合 V を m 個の部分集合 $V_1, V_2, \ldots, V_m$ に分割し、部分集合 V_i からの乗車要求をすべてバス B_i に割当てる方法。

GetOff: GetOn と同様の頂点集合の分割を用い、部分集合 V_i への降車要求をすべてバス B_i に割当てる方法。

Closest: 乗客の降車ポイントに最も近い車両に割り当てる方法。

5 シミュレーション

上述の 5 種類のオンライン配車方法の違いがシステム全体の性能に与える影響を評価するために、シミュレーション実験をおこなった。なお本提案システムでは、ディスパッチャによって割り当てられたインスタンスに対して実行可能解が存在しない場合、各車両ルータは与えられた要求をディスパッチャに戻すことになるが、今回の実験では配車方法自身の性能を純粋に比較するため、各要求がもつ (初期) デッドラインはすべて同一とし、割り当てられたインスタンスが実行可能解をもたない場合には、インスタンスが実行可能解をもつようになるまで全要求のデッドラインを均一に大きくするという方法をとった。したがって以下では、1) 移動距離の総和と 2) (初期) デッドラインの達成率のほか、3) 各乗客が乗車ポイントに到着してから目的ポイントに到着するまでの時間 (以下、**待ち時間**) の平均値に関しても各配車方法の性能を評価していくことにする。

5.1 パラメータ

実験で用いた設定は以下の通りである。まずバスの台数を 2 台に限定し、各バスの定員はいずれも 20 名とした。実験では、図 1 に示す頂点数 15 の地図を用いた。ここで地図中の 2 頂点間のユークリッド距離がその頂点間の距離にそのまま対応しており、各頂点間の距離は、例えば頂点 1 と 2 の間で 23 のように設定さ

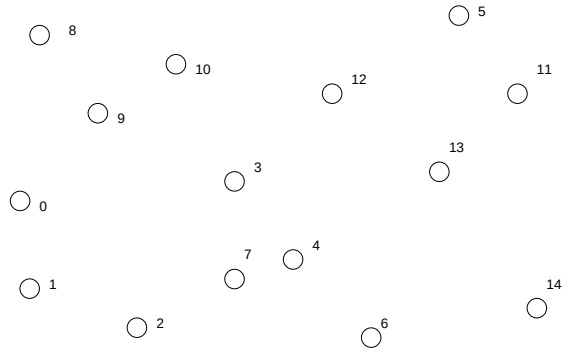

図 1: 実験で使用した地図.

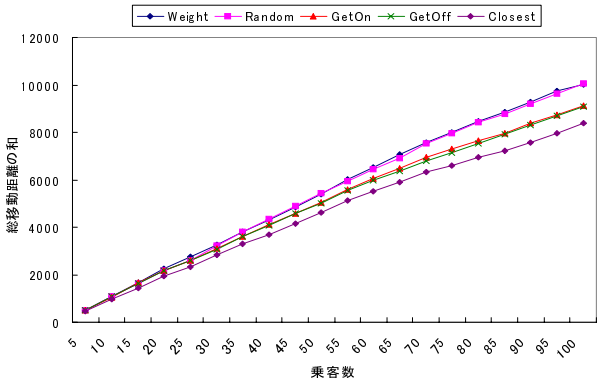

図 2: 総移動距離の和.

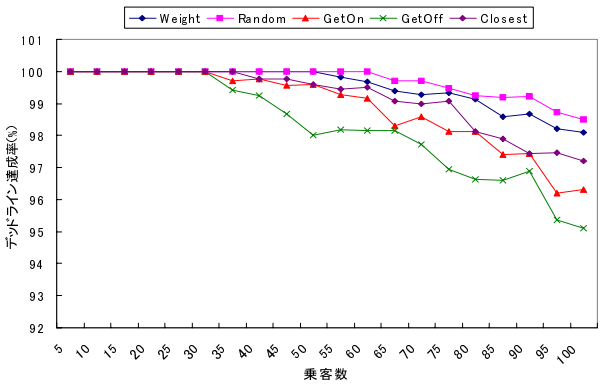

図 3: 初期デッドライン達成率.

れている[7]。また各ポイントでの乗降車時間は一律に $\Delta = 5$ とした。すべての要求系列に関して、全車両は初期状態において 0 番の頂点に存在するものとする。また配車方法 GetOn, GetOff で使用される頂点集合の分割は $\{\{1,2,3,7,8,9,10\},\{4,5,6,11,12,13,14\}\}$ とした。

乗客からの要求は 9000 単位時間 (約 15 時間に対応) の間にランダムに発生するものとする。ただしシミュレーション自体は、9000 単位時間が経過した後もすべての乗客が目的ポイントに到着し終わるまで続けられる。各利用者の乗車ポイントと降車ポイントは、グラフの頂点集合からそれぞれランダムに選択される。また各乗客の指定するデッドラインは、それぞれ乗車ポイント到着後 300 単位時間後に設定した (この値は約 30 分に対応している)。

5.2 結果および考察

総移動距離に関する結果を図 2 に示す。ここで縦軸は全要求に対する総移動距離 (の和) であり、横軸は 9000 単位時間中に発生した利用者数である。図よりあきらかに、Closest の性能がもっともよく、GetOff, GetOn, Random, Weight の順に総移動距離が伸びていることがわかる。 これは最初の 3 方式では (残りの 2 方式と違って) バスと乗降車ポイントに関する地理情報を直接利用しているためであると考えられる。ただし GetOff, GetOn では頂点集合の分割が静的になされているのに対して Closest では分割が動的になされており、その違いがわずかながら性能の差となってあらわれていることがわかる。

次に初期デッドラインの達成率について調べた。図 3 に結果を示す (図の横軸は図 2 と同様)。図よりあきらかに、初期デッドラインの達成率に関しては Random がもっともよく、Weight, Closest, GetOn, GetOff の順に性能は徐々に悪化していくことがわかる。なお GetOff や GetOn で性能が極端に悪化する理由は、頂点集合の静的な分割により乗車人数のアンバランスが起こりやすいためであると考えられる。また Closest では動的な分割をおこなうことで運行距離の短縮が図られており、その分デッドライン制約が満たされやすくなっていると考えられるが、それでも乗客の発生の具合によって負荷のアンバランスが生じるため、デッドライン制約の充足率は Random に比べるとあきらかに低下している。

最後に待ち時間の平均について評価した。結果を図 4 に示す。図を見る限り、この評価基準においても Closest がもっともよい性能を示し、Weight, Random でやや待ち時間が長くなり、GetOff や GetOn では待

[7] 前述のように、本稿では距離 1 を移動するのに要する時間を 1 単位時間と定義する。またここでは、10 単位時間がほぼ 1 分に対応すると考えている。

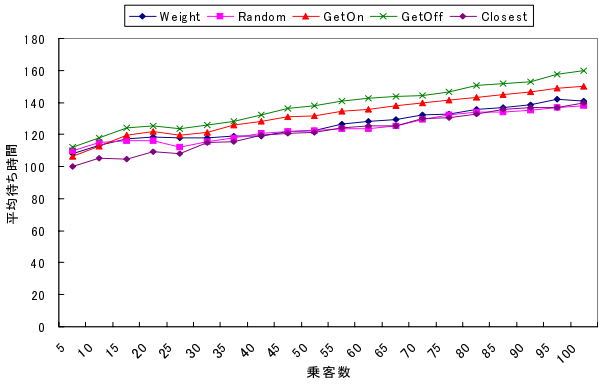

図 4: 移動時間の平均値.

ち時間はさらに長くなることがわかる。これは上記のデッドライン達成率で述べたように、GetOff や GetOn では負荷のアンバランスが起きやすいためであると考えられる。

上記の結果から、総移動距離、平均待ち時間の両面で Closest がもっともよい性能を示すことがわかる。また初期デッドラインの達成率でも比較的よい結果が得られていることから、ディスパッチャで用いられる配車方法としては、Closest に基づいた方法を採用するのが適切であると考えられる。

6 おわりに

本稿では、1 台の車両に対する運行距離最小化問題を解くための厳密解法を提案し、その解法をデマンドバスのためのオンライン配車システムに適応する方法について述べた。また提案する各種配車方式のよさについて、シミュレーションにより実験的に評価した。実験の結果、乗車ポイントにもっとも近いバスを割当てる方法が総移動距離と平均待ち時間の意味で優れていることがあきらかになった。

今回提案した最適解法では、A* の部分に最小生成木のサイズという比較的単純な評価関数を用いているが、DSP の性質をうまく利用することで、さらなる改良が可能であると考えている。また同時に配車方式についても、Closest を上回る性能をもった新しい配車方式の提案をおこなっていきたい。

参考文献

[1] F. H. Cullen, J. J. Jarvis, and H. D. Ratliff. Set Partitioning Based Heuristics for Interactive Routing. *Networks*, 11:125–143, 1981.

[2] J. Desrosiers, Y. Dumas and F. Soumis. A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows. *American Journal of Mathematical and Management Science*, 6:301–325, 1986.

[3] Y. Dumas, J. Desrosiers, and F. Soumis. The Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research*, 54:7–22, 1991.

[4] M. Fischetti and P. Toth. An additive bounding procedure for combinatorial optimization probolems. *Operations Research*, 37(2):319–328, 1989.

[5] B. Kalantari, A. V. Hill, and S. R. Arora. An algorithm for the traveling salesman problem with pickup and delivery customers. *European Journal of Operational Research*, 22:377–386, 1985.

[6] 中丁和也, 宮本俊幸, 熊谷貞俊. マルチエージェントネットを用いたデマンドバスシミュレーションシステム. 信学技報, CST2002-6:23–29, 2002.

[7] H. Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154, 1980.

[8] H. Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17(3):351–357, 1983.

[9] 内村圭一, 斉藤隆司, Hiro Takahashi. 遺伝的アルゴリズムによる乗客輸送の最適化. 電学論 (D), 117-D(7):891–897, 1997.

[10] 内村圭一, 斉藤隆司, Hiro Takahashi. 公共交通サービスにおける Dial-a-Ride 問題. 信学論, J81-A(4):599–606, 1998.