時間枠及び乗車時間ペナルティ付き乗合タクシー問題に対する反復局所探索法

竹田 陽

2020 年 10 月 26 日

Abstract. The dial-a-ride problem (DARP) consists of designing vehicle routes and schedules for multiple users who specify pickup and delivery requests between origins and destination. The aim is to design a set of minimun cost vehicle route while accommodating all requests. Side constraints include vechile capacity, route duration, maximum ride time, etc. DARP arises in share taxi service or transporting people in health care service. A number of papers about DARP has been published, however, most of the previous research deal with time windows and maximum ride time as hard constraints. In this study, we consider with the DARP where time windows and maximum ride time are soft constraints. These soft constraints make the DARP more generic. We propose several approaches to find optimal service times at each visiting point.

1 はじめに

近年，新しい移動手段として乗合タクシーサービスの需要が増加してきている．また，老人や介護が必要な人を自宅などからヘルスケアセンターなどまで輸送するようなサービスも，高齢化に伴い需要が増加してきている．都心部では，乗合タクシーの実証実験がさかんに行われており，実用化が進んできている．これらのサービスでは，サービス提供者はなるべく多くの顧客のリクエストを実行しつつ，サービスにかかるコスト (運行時間やルートの総距離) を少なくすることが重要となる．一方で利用者は，希望通りの時間にサービスを利用して，なるべく短い時間で出発地から目的地に着くことが重要である．つまり，サービス提供者と利用者が互いに満足のいくルートを作成することが必要である．

乗合タクシーサービスにおいて，利用者の満足度を考慮しつつサービスの実行にかかるコストを最小化する問題は乗合タクシー問題 (dial-a-ride problem, DARP) と呼ばれる．乗合タクシー問題の制約としては，車両の乗車人数の上限 (容量制約)，車両が回るルートの長さの上限 (最大ルート距離)，顧客が車両に乗っている時間の長さの上限 (最大乗車時間) が与えられる．

乗合タクシー問題に対しては，多くの先行研究があり，様々な手法が提案されている．また，複数のデポや異種車両などのさまざまな制約を考慮する研究もある．しかし先行研究では，乗降時刻を何時から何時までという時間枠で定めている．また，乗車時間に関しても何分までなら許容できるという最大乗車時間で定めていることが一般的である．

本研究では，制約として時間枠及び乗車時間に対するペナルティ関数を与えてソフト制約とする乗合タクシー問題について考える．ペナルティ関数は区分線形凸関数であるとする．区分線形関数にすることで，ルート全体の効率を上げるために多少の遅延を許容できる場合など，様々なケースを柔軟に考慮することができるようになる．また，乗車時間の増加につれてペナルティの値も大きくなるので，顧客の不満度をより柔軟に表現することが可能となる．これらの点より，時間枠及び乗車時間ペナルティ付き乗合タクシー問題は先行研究より汎用的な問題であると言える．

DARP に関して，既存研究をいくつか紹介する．Jaw らは，この問題に対して近似解法としてルートに挿入した時の目的関数値の増加が最小になるようなリクエストを選択してルートに挿入していく連続挿入法を提案した [3]．Cordeau らは，あるルートか

らリクエストをひとつ取り除き，別のルートに挿入する際にタブーサーチ探索を用いる手法を提案した [2]．Braekers らは，DARP に対しての発見的解法として焼きなまし法 (simulated annealing，SA) を用いた手法を提案した [1].

2 問題説明

まず，pickup and delivery problem (PDP) について説明し，そのあとに乗合タクシー問題 (DARP) について詳しく説明する．

2.1 pickup and delivery problem

pick and delivery problem (PDP) は，荷物を出発地まで目的地まで輸送する問題である．PDP では，荷物を受け取る地点 (pickup) と荷物を配送する地点 (delivery) のペアからなる n 個のリクエストと m 台のが与えられたときに以下の制約を満たしつつ与えられたコストを最小化するルートを求める問題である．

1. 与えられた n 個のリクエスト全ての地点を訪問する．
2. 全ての車両はデポから出発してデポに帰る．
3. リクエストでペアになっている出発地と目的地は同じ車両が訪問する．
4. それぞれのリクエストにおいて，必ず出発地を訪問した後に目的地を訪問する．

ここでのリクエストとは，荷物の輸送要求のことを指す．

与えられるコストとしては各頂点間の距離や時間などがある．よく制約として考えられるものは，リクエストの訪問点を決められた時間内に訪れなければならない時間枠制約や車両の容量を制限した容量制約などがある．また，車両の種類が複数あり，車両によって最大容量が異なる多資源制約や，複数のデポを考慮する問題など，様々な拡張が考えられている．

2.2 dial-a-ride problem

乗合タクシー問題 (dial-a-ride problem，DARP) は，PDP を人の輸送に特化した問題である．DARP は PDP と違い人を輸送するため，車両に乗っている乗車時間が長いと利用者の不満がたまってしまう．そこで乗車時間やリクエストの訪問時間のずれなどで評価される不満度を考慮する必要がある．

顧客の満足度は，利用者の乗車時間と出発地と目的地での待ち時間ではかる．本研究では，利用者が乗車時刻，降車時刻，乗車時間のそれぞれに対して希望を持つことを考える．それぞれの希望はそれぞれ連続区分線形凸関数のペナルティ関数で表される．ペナルティ関数を区分線形関数にすることで，任意の間隔ごとに関数を設定することが可能になり，3 分の遅延なら許容できるが 20 分の遅れは許容できないなどの表現が可能になる．また，乗車時間に応じてペナルティをかけることができるので，不満度を柔軟に表現することが可能になる．したがって，DARP をより汎用的に解くことが可能となる．

事前に全てのリクエストがわかっている問題を静的 DARP，リクエストが全てはわかっておらず問題を解く過程でリクエストが次々と与えられる問題を動的 DARP といい，本研究では静的 DARP を考える．

2.3 乗合タクシー問題の問題複雑度

DARP は，使用する車両数を 1，リクエストにおける出発地を全てデポとすることで巡回セールスマン問題 (traveling salesman problem，TSP) に帰着することができる．TSP は NP 困難 [4] であることが知られているため，DARP も NP 困難である．

2.4 定式化

乗合タクシー問題は，$G = (V, E)$ の完全有向グラフ上で定義される．$V = \{\{0, 2n+1\} \vee P \vee D\}$ を頂点集合とし，$E = \{(i, j) \mid i, j \in V, i \neq j\}$ を各頂点間の辺集合とする．ここで，デポを 0 と $2n+1$ で表し，V の部分集合 $P = \{1, \ldots, n\}$ を乗車地点の集合，V の部分集合 $D = \{n+1, \ldots, 2n\}$ を降車地点の集合とする．$i \in P$，$i + n \in D$ に対して，V 上の 2 点のペア (i , $i + n$) は乗車地点の頂点 i から目的地に対応する頂点 $i + n$ への乗客輸送リクエスト (以下，単にリクエストと呼ぶ) とする．それぞれの頂点 $v_i \in V$ には，負

荷 q_i とサービス時間 s_i が与えられる．

各リクエストは，m 台の車両 $k \in K = \{1, 2, \ldots, m\}$ で訪問される．σ をルートの集合とし，σ_k を車両 k の訪問する頂点の順列とすると，$\sigma = \{\sigma_1, \ldots, \sigma_m\}$ である．また，$\sigma_k(h)$ は車両 k が h 番目に訪問する頂点をあらわす．頂点 i，$j \in V$ 間の距離を c_{ij}，時間を t_{ij} とする．車両の容量を Q とし，車両 k の $\sigma_{k(h)}$ における容量を $q_{\sigma_k(h)}$ とする．x_{ij} を 0-1 変数とし，頂点 i，j を結ぶ辺がルートに含まれれば 1，そうでなければ 0 とする．各車両は 1 つのデポから出発しデポに帰る．ここで，n_k は車両 k が訪問するデポ以外の頂点の数である．

この問題に対して，目的関数として

1. 車両運用コスト最小化，
2. 顧客満足度最大化

の 2 つを考える．車両運用コストとしては，様々なコストが考えられるが本研究では車両が走行するルートの総距離とする．ルートの総距離を $d(\sigma)$ とすると，

$$d(\sigma) = \sum_{k \in K} \sum_{h=0}^{n_k} c_{\sigma_{k(h)}, \sigma_{k(h+1)}}$$

と表せる．各リクエストにおける乗車時刻に対するペナルティ関数を g_i^+，降車時刻に対するペナルティ関数を g_i^-，乗車時刻に対するペナルティ関数を g_i とする．頂点 i でのサービス開始時刻を τ_i とし，I をリクエスト集合として，I_σ をルート σ に属するリクエストの集合とする．また，利用者の不満度を $t(\sigma)$ とすると，

$$t(\sigma) = \sum_{i \in I_\sigma} (g_i^+(\tau_i) + g_i^-(\tau_{i+n}) + g_i(\tau_{i+n} - \tau_i))$$

と表せる．各ルートについて車両の割り当てとリクエストの訪問順が決まっている 1 つのルートが与えられた場合に，各頂点でのサービス開始時刻を決定する必要がある．本研究では，目的関数と制約が全て線形の式で表すことが可能なので，線形計画問題 (linear programming problem，LP) として解くことができる．

本研究では，2 つの目的関数の重み付き和の最小化を考える．そうすることで，少し顧客の不満度が上がっても良いのでルートの総距離を短くしたい場合や，顧客の不満度を最優先で考えたい場合など，様々な状況を考慮することができる．

それぞれの係数を定数 α と β としたとき，以下のように定式化できる:

$$\text{minimize} \quad \alpha d(\sigma) + \beta t(\sigma), \tag{1}$$

$$\text{subject to} \quad \sum_{i \in V} x_{ij} = 1 \qquad (j \in P \cup D), \tag{2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad (i \in P \cup D), \tag{3}$$

$$\sum_{k \in K} n_k = 2n, \tag{4}$$

$$\sigma_k(0) = \sigma_k(n_k + 1) = 0 \qquad (k \in K), \tag{5}$$

$$0 < q_{\sigma_k(h)} < Q \quad (h \in \{0, ..., n_k\},\ k \in K), \tag{6}$$

$$g_i(\tau_{i+n} - \tau_i) > s_i + t_{i,\ i+n} \qquad (i \in I), \tag{7}$$

$$\tau_{\sigma_k(h+1)} \geqq \tau_{\sigma_k(h)} + s_{\sigma_k(h)} + t_{\sigma_k(h), \sigma_k(h+1)} \quad (h \in \{0, ..., n_k\}, k \in K). \tag{8}$$

式 (2)，(3) は訪問点が 1 回ずつしか訪問されないことを，式 (4) は全てのリクエストが実行されることを表している．式 (5) は全ての車両がデポから出発してデポに帰ることを表している．式 (6) は車両の容量制約を表している．式 (7)，(8) はリクエストの先行制約とサービス時刻に関する制約を表している．

3 提案手法

本節では，本研究で提案する手法について説明する．本研究では，リクエストの割り当てと訪問順を反復局所探索法を用いて求める．それぞれの反復で得られたルートに対しては，LP ソルバーを使って最適なサービス開始時刻を決定するアルゴリズムを提案する．

3.1 初期解生成

リクエストをランダムに選び，車両 k にリクエストのペアが連続となるようにルートの最後に挿入する．これを $k = 1$ から m まで繰り返した後，未割り当てのリクエストがあれば $k = 1$ として同様の操作を続ける．これを未割り当てのリクエストがなくなるまで続ける．このように生成することで，デポから出発してリクエスト全てを訪問し，同じ車両で出発地のあとに目的地を訪問するという制約を必ず守る初期解を生成することができる．

3.2 制限の緩和

本研究では，局所探索を行う上でより自由に探索を行うために，車両における容量制約を緩和し，容量制約を破った時のペナルティを計算するペナルティ関数を定義する．容量制約のペナルティ関数を目的関数に加えた評価関数を用いて解を評価することにより，実行不可能解も探索可能になる．

車両の容量制約のペナルティは，車両の最大容量を超えて乗った人数として，H と表す．車両 k に対してルートの i 番目を訪問後に容量を超えて乗っている人数を H_{ki} とすると，

$$H = \sum_{k \in K} \sum_{i \in n_k} H_{ki},$$

と表せる．γ を定数とすると，ペナルティを加えた評価関数を $f(\sigma)$ とすると

$$f(\sigma) = \alpha d(\sigma) + \beta t(\sigma) + \gamma H,$$

で定義する．

3.3 局所探索法

局所探索法 (local search) とは，解を逐次的に改善させていく手法である．解に変化を少し加える操作を近傍操作と呼び，近傍操作によって生成されうる解の集合を近傍と呼ぶ．局所探索法では，適当な初期解からはじめ，現在の解の近傍内に，より良い解が存在すればその解に移動する，という操作を近傍内に改善がなくなるまで反復する方法である．本研究では，近傍操作は大きく分けてルート内の操作とルート間の操作の 2 種類を用いる．

3.4 ルート内の近傍操作

1 つのルートの 1 つの頂点を選び，同じルート内の別の箇所に挿入し直す操作である．本研究では 1 つのルートで改善がなくなるまでルート内の近傍操作を行うが，探索する近傍サイズはルート内のデポを除いた頂点数を a とすると $2a^2$ とした．

3.5 ルート間の近傍操作

ルート間の近傍操作では，挿入近傍と交換近傍の 2 種類を実装し，計算結果を比較した．2 種類ともに，ルート間の近傍操作を行い，現在の解の評価関数よりも改善した場合，操作後の近傍解に移動する．

3.5.1 挿入近傍

1 つのルートから 1 つのリクエストペアを選び，別のルートに挿入する操作である．挿入する場所は評価関数が最も良くなる場所とする．

3.5.2 交換近傍

1 つのルートから 1 つのリクエストペアを選び，別のルートのリクエストペアと交換する操作である．挿入近傍と同じく，新たに挿入する場所は，最も評価関数がよくなる場所とする．

3.6 局所探索アルゴリズム

時間枠及び乗車時間ペナルティ付き乗合タクシー問題に対する局所探索アルゴリズムとして，まずルート内で近傍操作を改善がなくなるまで行い，得られた局所最適解に対してルート間の近傍操作を行い，最良の箇所に挿入する．改善している場合，局所最適解を更新し，現在の解から移動する．

以下に，x_{init} を初期解，x を現在解，x_{best} を暫定解とした際の提案手法の概要を示す．

3.7 最適なサービス開始時刻決定

本研究では，各頂点でのサービス開始時刻を線形計画問題として定式化を行い，Gurobi Optimizer (ver 9.0.0) を LP ソルバーとして使用している．しかし現在は探索 1 回ごとにサービス時刻を決定する LP を解いているため，計算に多くの時間を要している．この計算時間の長さによって，ヒューリスティックの手法のアプローチをとれていないため，3 つの Gurobi

Algorithm 1 局所探索法

1: $x := x_{\text{init}}$ とする．
2: 改善がなくなるまでルート内の近傍操作を行う
3: ルート間の近傍操作を行う
4: 変化があったルートに対してルート内の近傍操作を行う
5: **if** 改善した **then**
6: $\quad x_{\text{best}} := x$
7: **end if**
8: x_{best} を局所最適解として出力して終了．

Optimizer のモデルの持ち方を提案し，計算実験を行う．以下に 3 つを示す．

1. すべての車両の情報をひとつのモデルに保持し，探索ごとにすべての車両の各頂点においての時刻を再決定
2. 車両ごとにモデルを保持し，探索ごとに変更のあった車両の各頂点における時刻を再決定
3. すべての車両の情報をひとつのモデルに保持し，探索ごとに変更のあった頂点に対して時刻を再決定

4 計算実験

4.1 実験環境

実験に用いるプログラムは C++ を用いて実装し，計算機はプロセッサ 1.4 GHz Intel Core i5，メモリ 16 GB 2133 MHz LPDDR3 の macOS を搭載したものを使用した．

4.2 問題例の作成方法

DARP では多くの既存研究があるが，本研究では時間枠及び乗車時間に対して区分線形で凸のペナルティ関数で与えている．このような問題設定のインスタンスは存在していないため，ベンチマークとしてよく使用される Cordeau らによって提供されている [2] インスタンスに修正を加えて計算実験を行う．修正方法を以下に示す．時間枠に関しては，サービス開始可能時刻を e，サービス開始最遅時間を l とすると，0 以上 e 以下に対しては傾き-1，e から l に対しては値が 0，l 以上に対しては傾きが 1 となるような，区分数が 3 のペナルティ関数を作成する．この修正作業を全てのリクエストに対して行う．乗車時間に関しては，乗車時間の閾値を L とすると，0 以上 L 以下に対しては値が 0，L 以上に対しては傾きが 1 となるような区分数 2 のペナルティ関数を作成する．この修正作業を全てのリクエストペアに対して行う．

4.3 インスタンスについて

計算実験に使用するインスタンスは，以下の特徴を持つ．

1. 訪問点におけるサービス時間 d を 10 とする．
2. 乗降人数は 1 人とする．
3. 訪問点 v_i と v_j 間の距離 c_{ij} と時間 t_{ij} は，2 つの頂点のユークリッド距離とする．
4. 乗車時間の閾値 (最大乗車時間) を 90 とする．
5. 車両の最大容量を 6 人とする．
6. それぞれの車両のルートの最大の長さを 480 とする．

以上の特徴を持つリクエスト数 24 から 144 のインスタンスを使用して計算実験を行う．

4.4 実験結果

Gurobi Optimizer における 3 つのモデルの持ち方による計算時間及び目的関数の最良値を表 1 に示す．近傍操作には，挿入近傍を用いた．また，目的関数におけるペナルティを $\alpha = 1, \beta = 500$ とする．

3 つの異なるモデルの持ち方を実装したが，計算時間に大きな違いは見られなかった．また，ルートの総距離とペナルティの違いに関しては，実装のランダム性における局所最適解の精度の差異だと考えられる．

5 まとめと今後の課題

時間枠及び乗車時間ペナルティ付き乗合タクシー問題に対して，反復局所探索法による解法を提案した．今後の研究計画としては，kick の方法などについて考察していきたい．また，より精度の良い解を

得るために，近傍操作の見直しなどを行っていく．

参考文献

[1] Kris Braekers，An Caris，and Gerrit K Janssens. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. Transportation Research Part B: Methodlological，Vol. 67，pp. 166–186，2014

[2] Jean-François Cordeau and Gilbert Laporte. A Tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B: Methodological. 37. 579-594. 2003.

[3] Jang-Jei Jaw，Amedeo R Odoni，Harilaos N Psaraftis and Nigel H. M. Wilson A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research Part B: Methodological. Vol. 20，No. 3，pp. 243-2435，1986.

[4] 柳浦睦憲，茨木俊秀，組合せ最適化 メタ戦略を中心として，朝倉書店，2001

表 1　モデルの実装方法の比較

	パターン 1		パターン 2		パターン 3	
問題例	総距離	計算時間	総距離	計算時間	総距離	計算時間
r1a	219.27	117.14	237.30	154.22	232.51	105.23
r1b	208.56	183.03	254.11	124.93	232.51	105.23
r2a	431.61	847.93	437.01	1136.47	437.98	877.47
r2b	402.11	843.17	465.56	1311.07	421.96	1110.43
r3a	779.04	3404.06	825.01	3901.42	806.99	3155.24
r3b	697.87	3231.22	817.24	3948.20	767.10	3674.59