Szeto,W.Y., Jiang,Y.

Transit route and frequency design: Bi-level modeling and hybrid artificial bee colony algorithm approach

Transportation Research Part B, Vol.67, pp.235-263, 2014.

理論談話会 2014/6/28 -Sat

吉野 大介

目次

- Introduction
- 二段階問題
- アルゴリズム
- ケーススタディ
  - 小規模ネットワーク
  - 現実ネットワーク

Introduction

■既往研究の課題

- これまでの公共交通計画では，ルート設計と便数設計をそれぞれ別々に扱ってきたが，両者はいずれも公共交通のLOS（車内混雑や停留所での待ち時間等）を決定する指標であり，同時に解く必要性が高い.
- 乗客の利便性を大きく左右する指標として「車内混雑」と「乗換回数」が挙げられるが，モデル内で明示的に取り扱っている事例がない.

■本研究の貢献

- 二段階問題によって公共交通のルート・便数の設計を同時に行う.
- 目的関数及び制約条件内で車内混雑・乗換回数を明示的に取り扱う.
- モデルを解くためのアルゴリズムとしてHybrid ABCアルゴリズムを採用.
- ケーススタディによりモデル・アルゴリズムの適用可能性について検証.

二段階問題

■上位問題：便数／ルート計画

- ネットワーク全体での乗換客数合計値の最小化が図られるような各路線の便数とルートを決定する（制約条件により，同時に停留所の位置も決まる）．

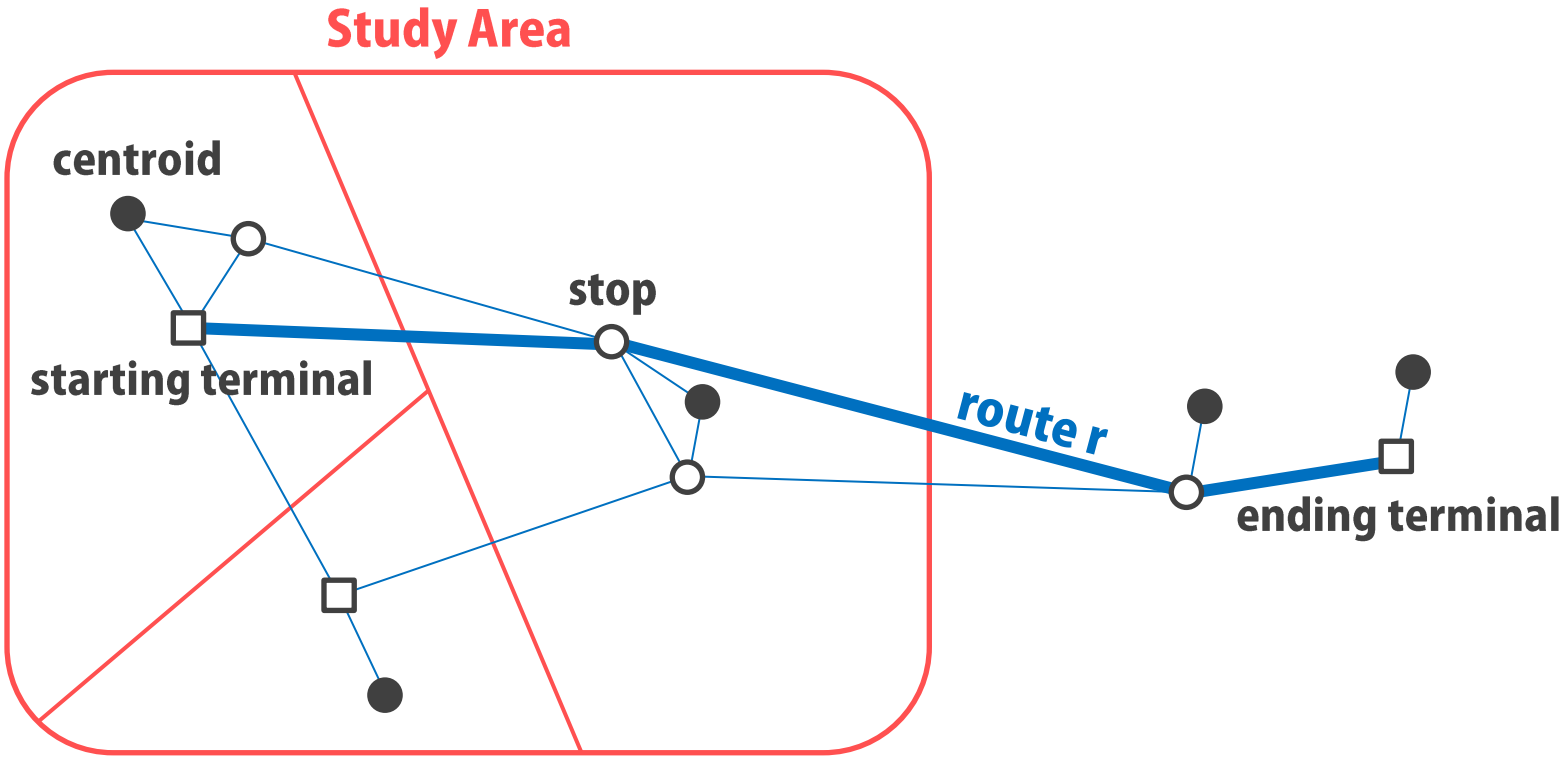
■下位問題：乗客配分計画

- 上位問題によって決定したルートに基づき，乗客配分を行う．
- 配分計算の際の乗客のルート選択行動については，乗車時間・停車時間・待ち時間の合計を最小化するルートを選択するとともに，リンク・車両の容量制約を考慮する．

上位問題

■上位ネットワーク

- ノード（停留所＋セントロイド）とリンクによる有向グラフでネットワーク形成.
- エリアはゾーンに分割し，各ゾーンにセントロイドmを配置．OD需要量はセントロイド間で付与．各セントロイドは同一ゾーン内の停留所候補地及びターミナル候補地と接続.
- 起点ターミナルeはStudy Area内，終点ターミナルe'はStudy Area外に設置.



上位問題

X_{0jr}：ルートrがノードjから開始している場合1,そうでなければ0,
X_{i0r}：ルートrがノードiで終了している場合1,そうでなければ0,
X_{ijr}：ルートrがノードiの直後にノードjを通過する場合1,そうでなければ0,
v_t^e：乗換リンクtで乗り換えて目的地eに向かう乗客数,
c_{ij}：ノードiからノードjまでの最短経路の乗車時間，s_t：ノードにおける平均停車時間,
T_r：ルートrにおける起点～終点までの旅行時間.

■目的関数

$$\min_{\mathbf{x},\mathbf{f}} z_1 = \sum_{t\in T^R}\sum_{e\in G_d} v_t^e, \quad \text{(1) 全ルートにおける乗換者数} v_t^e \text{の合計値を最小化}$$

■制約条件

$$\sum_{j\in U\cup\{0\}} X_{0jr} = 1 \quad \text{for} \quad r = 1 \text{ to } R_{\max}, \quad \text{(2) 全ルートとも起点バスターミナルから出発}$$

$$\sum_{i\in V\cup\{0\}} X_{i0r} = 1 \quad \text{for} \quad r = 1 \text{ to } R_{\max}, \quad \text{(3) 全ルートとも終点バスターミナルに到着}$$

$$\sum_{\substack{i\in Z_U\cup\{0\}\\ i\neq i}} X_{ijr} - \sum_{\substack{i\in Z_U\cup\{0\}\\ i\neq i}} X_{jir} = 0 \quad \text{for} \quad j\in Z_U,\ r = 1 \text{ to } R_{\max}, \quad \text{(4) 起終点以外についてはどのノードについても前後のノードが存在}$$

$$\sum_{\substack{i\in Z_U\cup\{0\},\\ i\neq j}} X_{ijr} \leqslant 1 \quad \text{for} \quad j\in Z_U,\ r = 1 \text{ to } R_{\max}, \quad \text{(5)}$$

$$\sum_{\substack{j\in Z_U\cup\{0\},\\ j\neq i}} X_{ijr} \leqslant 1 \quad \text{for} \quad j\in Z_U,\ r = 1 \text{ to } R_{\max}, \quad \text{(6)}$$

$$X_{jjr} = 0 \quad \text{for} \quad j\in Z_U,\ r = 1 \text{ to } R_{\max}, \quad \text{(7)}$$

(5)(6)(7)：各ノードはどこかのルートにおいて必ず1回は通過される

$$T_r = \sum_{i\in Z_U}\sum_{j\in Z_U, j\neq i} X_{ijr}\left(c_{ij} + s_t\right) - s_t \quad \text{for} \quad r = 1 \text{ to } R_{\max}, \quad \text{(8) 各ルートの乗車時間} T_r \text{の算出}$$

上位問題

X_{00r}：ルートrが利用不可能な場合1,そうでなければ0,
f_r：ルートrの便数，f_{min}：最小便数，S_{max}：最大中間停留所数，T_{max}：最大旅行時間数，
k_{cap}：バス容量，d_m^e：ノードmからセントロイドeへの需要，
δ_r^e：ルートrが目的地eに接続されている場合1，そうでなければ0，
q_{ir}：サブツアー排除のためのノードiに設定する定数項，p：サブツアー排除のための巨大数.

■制約条件（続き）

$$\sum_{r=1}^{R_{\max}} 2 f_r T_r (1 - X_{00r}) \leqslant W,$$

(9) 全ルートの運用車両数は実現可能車両W以下

$$f_{\min}(1 - X_{00r}) \leqslant f_r \quad \text{for} \quad r = 1 \text{ to } R_{\max},$$

(10) ルートの便数は最小便数f_{min}以上

$$\sum_{i \in C} \sum_{j \in C, j \neq i} X_{ijr} \leqslant S_{\max} \quad \text{for} \quad r = 1 \text{ to } R_{\max},$$

(11) ルートの中間停留所数は最大停留所数S_{max}以下

$$\sum_{i \in C} \sum_{j \in C, j \neq i} X_{ijr} (c_{ij} + s_t) - s_t \leqslant T_{\max} \quad \text{for} \quad r = 1 \text{ to } R_{\max},$$

(12) ルートの旅行時間は最大旅行時間T_{max}以下

$$\sum_{r=1}^{R_{\max}} \sum_{i \in H_m} \sum_{j \in Z_U \cup \{0\}, j \neq i} X_{ijr} \geqslant 1 \quad \text{for} \quad m \in G_s,$$

(13) ゾーンカバー制約（各ゾーン内において1つ以上の路線がどこか1つの停留所に停止する）

$$\sum_{r=1}^{R_{\max}} f_r k_{\text{cap}} \delta_r^e \geqslant \sum_{m \in G_s} d_m^e \quad \text{for} \quad e \in G_d,$$

(14) 容量制約（車両容量×頻度の合計は需要の合計以上）

$$\delta_r^e = \sum_{i \in Z_U \setminus V e' \in H_e} \sum X_{ie'r} \quad \text{for} \quad e \in G_d, \text{ and}$$

(15) ルートrはターミナルe'で終了する

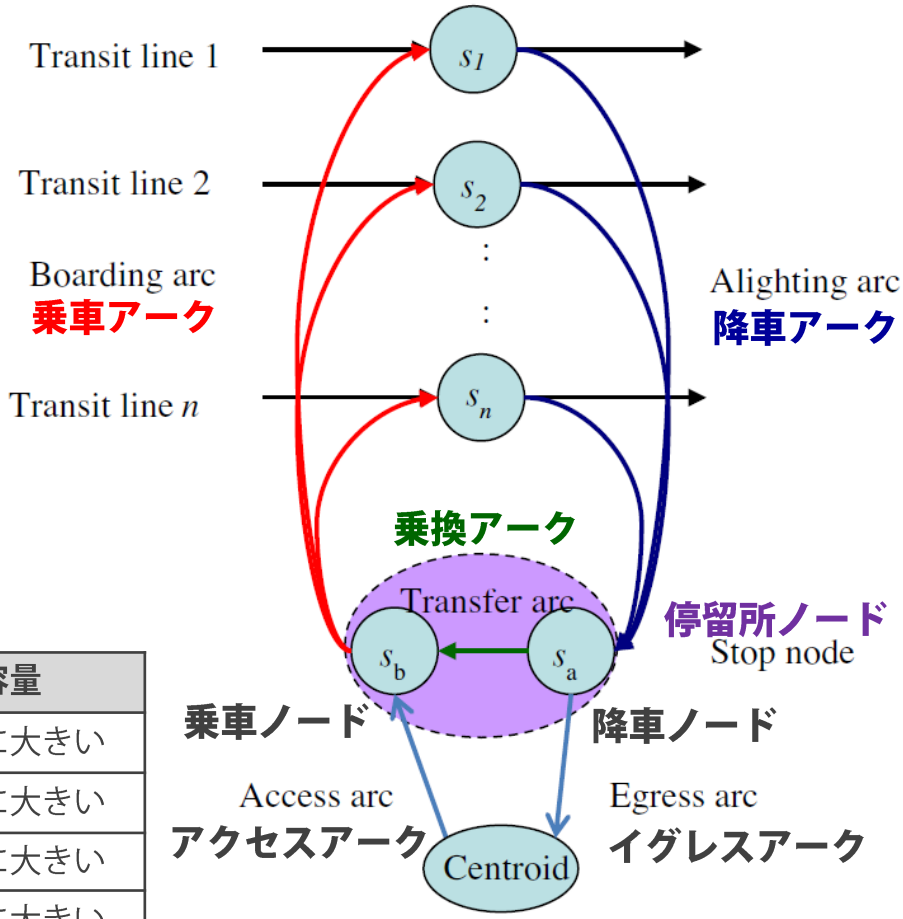
$$q_{ir} - q_{jr} + p X_{ijr} \leqslant p - 1 \quad \text{for} \quad i, j \in Z_U, \quad i \neq j, \quad r = 1 \text{ to } R_{\max}.$$

(16) サブツアー排除制約

下位問題

■下位ネットワーク

- バス路線とセントロイドの接続をアークで表現．各アークは移動時間・便数・容量の情報を持つ．
- 乗車アークと降車アークが各路線$s_1 \cdots s_n$と上位問題の停留所ノードを連結．
- 停留所ノードは降車ノードs_aと降車ノードs_bで表現，乗換アークが両者を連結．
- アクセスアーク・イグレスアークはセントロイドから停留所ノードまでの徒歩移動に該当．
- 停留所と停留所の間はトラベルアークで連結．



	移動時間	便数	容量
乗車アーク	なし	公共交通の便数	非常に大きい
降車アーク	なし	非常に大きい	非常に大きい
乗換アーク	不便費用Mを設定	非常に大きい	非常に大きい
アクセスアーク	徒歩時間	非常に大きい	非常に大きい
イグレスアーク	徒歩時間	非常に大きい	非常に大きい
トラベルアーク	乗車時間 ＋停留所停車時間	公共交通の便数	バス容量×便数

下位問題

c_a：リンクaの乗車時間，
v_a^e：リンクaから目的地eに向かう乗客数，
ω_i^e：ノードiにおける目的地eに向かう流動の総待ち時間，
f_a：リンクaにおける便数，
A：下位ネットワークのリンク集合，A_i^+：ノードiから出ていくリンク集合．

■目的関数

$$\min_{\mathbf{v},\mathbf{w}} : z_2 = \sum_{a\in A}\sum_{e\in G_d} c_a v_a^e + \sum_{i\in Z_L}\sum_{e\in G_d} \omega_i^e, \quad \text{(17)乗車時間と待ち時間の合計を最小化}$$

■制約条件

$$v_a^e \leqslant f_a \omega_i^e \quad \text{for} \quad a \in A_i^+,\ i \in Z_L,\ e \in G_d, \quad \text{(18) 流動量・便数・待ち時間に関する制約}$$

$$\sum_{a\in A_i^+} v_a^e = \sum_{a\in A_i^-} v_a^e + d_i^e \quad \text{for} \quad i \in Z_L,\ e \in G_d, \quad \text{(19) ノードにおける流動量保存制約}$$

$$\sum_{e\in G_d} v_a^e \leqslant f_a k_{\text{cap}} \quad \text{for} \quad a \in A, \quad \text{(20) トラベルアークの流動量} v_a^e \text{はアーク容量} f_a k_{cap} \text{よりも小さい}$$

$$V_a^e \geqslant 0 \quad \text{for} \quad a \in A, e \in G_d, \text{ and} \quad \text{(21)}$$

$$\omega_i^e \geqslant 0 \quad \text{for} \quad i \in Z_L,\ e \in G_d. \quad \text{(22)}$$

} 非負制約

アルゴリズム

■ヒューリスティクスの適用

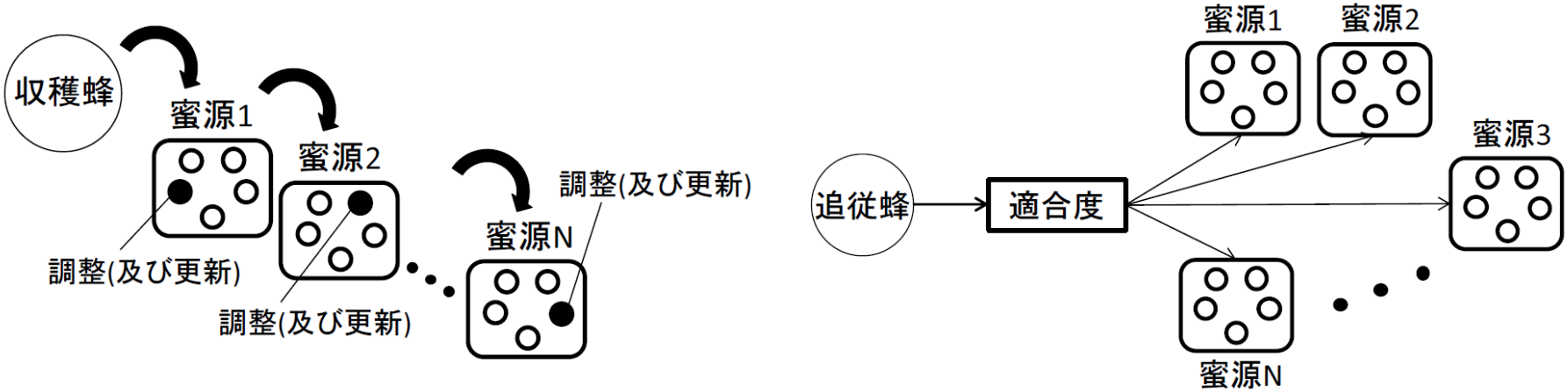
- 上位問題は混合整数非線形計画問題であり二段階問題はNP困難.
- メタヒューリスティクス手法のひとつであるABCアルゴリズムを採用.
- ただし，制約条件が多すぎるため二段階問題にABCアルゴリズムを直接適用することはできない.
- Hybrid ABCアルゴリズムを適用.
  - ルート設計問題：ABCアルゴリズム
  - 便数決定問題：便数決定アルゴリズム（Frequency determination algorithm）

  ※下位問題は便数決定問題の中のステップで計算

ABCアルゴリズムの概要

■ABCアルゴリズムとは

- 蜜蜂の群れによる採餌行動にヒントを得た最適化アルゴリズム．餌場と三種類の蜜蜂群から構成されており，以下の3つの作業をまとめて反復1回として計上．
- 調整＝解の計算のみ，更新＝計算した上で解を上書きすること．
  - 働き蜂（Employed bee）：餌場全てを通過し，中身である解をランダムで一つ選んで調整する．もし調整した解が最良解と比較して良くなれば解を更新，悪くなったら更新しない．働き蜂は餌場の数と同数配置される．
  - 見学蜂（Onlooker bee）：見物蜂は働き蜂の探索による情報に基づき，相対的に評価（適応度）の高い餌場を一つ選び，餌場内の解の調整を行う．
  - 偵察蜂（Scout bee）：一定回数働き蜂にも見学蜂にも解が更新されなかったら，働き蜂は偵察蜂に変化し，その餌場の中から解をランダムに選びなおす．

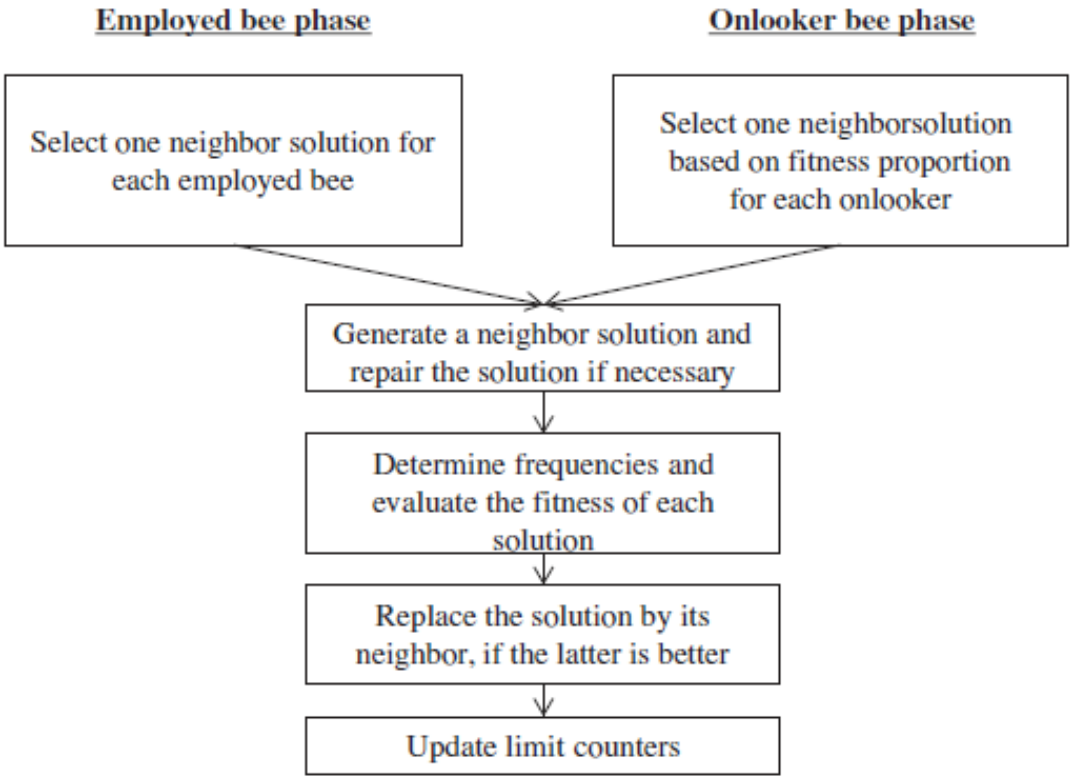


図面出典：小澤優太，坂本佑樹，大用庫智，高橋達二：ミツバチコロニーアルゴリズムにおける情報活用と探索の混同とその機能，2013年度人工知能学会全国大会（第27回）予稿集，2013.

ABCアルゴリズムの概要

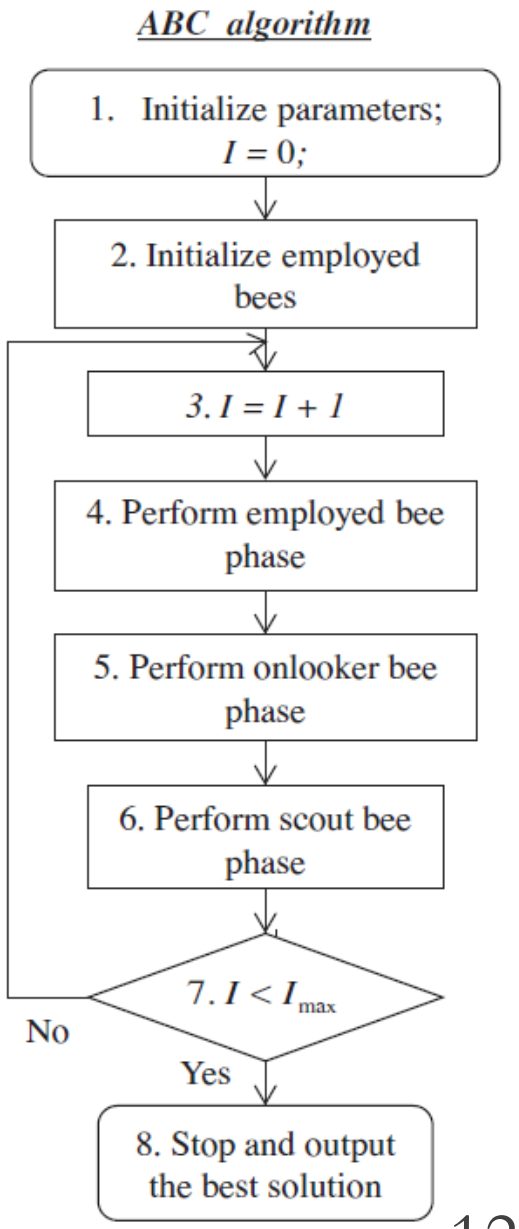
■働き蜂と見物蜂のフェーズの違い

- 働き蜂は餌場全てを通過し，中身である解をランダムで1つずつ選び調整（近傍探索）する．もし調整した解が最良解と比較して良くなれば解を更新，悪くなれば解を更新しない．
- 見物蜂は餌場を適合度によって選ぶ．餌場ごとに適合度を持っており，各餌場が持つ解が最適解とどれだけ合っているかどうかを示す値．適合度が高い餌場はより多くの見物蜂が訪れ，働き蜂と同じ要領で解の調整・更新が行われる．



ABCアルゴリズムの流れ

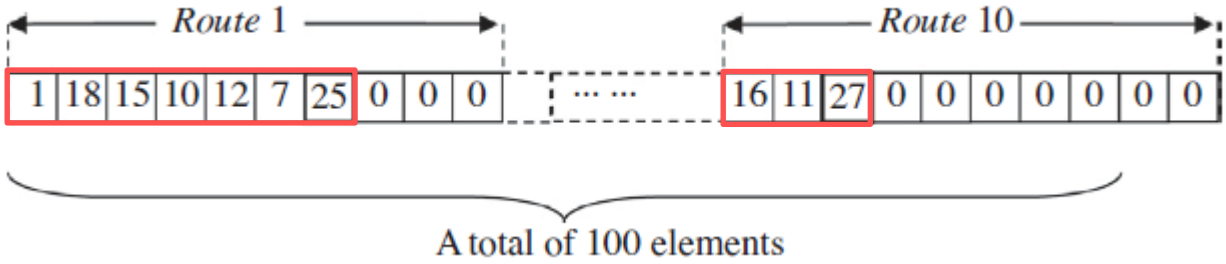
- **Step1: 初期設定**
  - コロニーサイズN_c，働き蜂数N_e， 見物蜂数N_o，偵察蜂数N_s，働き蜂が偵察蜂に変化する閾値$limit$，総反復回数I_{max}
  - 反復回数$I = 0$から開始
- **Step2: 初期化**
  - 働き蜂を餌場にランダム配置し初期配置における最良解を算出
- **Step3: 反復回数を1増やす**
- **Step4: 働き蜂フェーズ**
  - 近傍探索により前期の最良解に対して適応度が高い近傍解があれば入れ換えて働き蜂の未更新回数を0にリセット，なければ前期最良解を維持し，未更新回数を1増やす
- **Step5: 見学蜂フェーズ**
  - 適応度に応じて相対確率を算出，相対確率に基づくルーレット選択によって探索点を選択し，Step4と同様に前期解と比較
- **Step6: 偵察蜂フェーズ**
  - $limit$以上連続して更新されなかった働き蜂はその餌場の中からランダムに解を選びなおす
- **Step7: 終了判定（$I < I_{max}$ならStep3へ）**
- **Step8: 最適解出力**



ABCアルゴリズムによる解の生成・修復

■ABCアルゴリズムにおける解の表現方法

10 elements × 10 routes = 100 elementsで表現



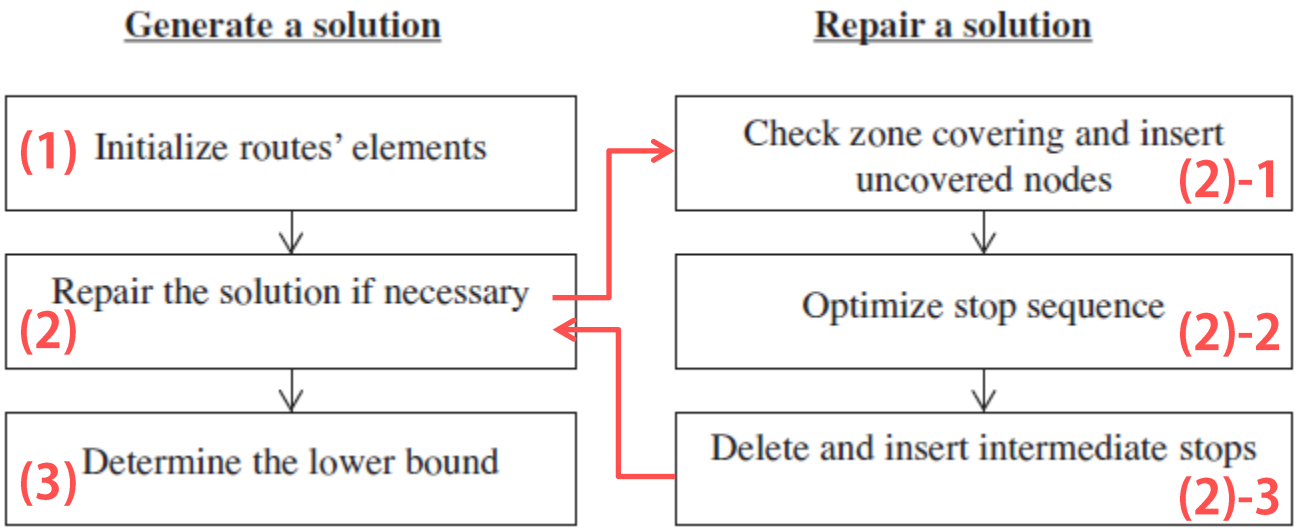
Route 1
起点ターミナル1⇒中間停留所18,15,10,12,7,25⇒終点ターミナル25

⋮

Route 10
起点ターミナル16⇒中間停留所11⇒終点ターミナル27

ABCアルゴリズムによる解の生成・修復

■解の生成と修復手順



(1) routes' elementsの初期化

- ABCアルゴリズムの初期化及び偵察蜂フェーズに該当.
- 各ルートに対し，起点ノードは起点ターミナルからランダム選択，最終ノードは終点ターミナルからランダム選択される.
- 中間停留所はターミナル間に挿入される．停留所ノードiの選択可能性$p_i = d_i^{e\prime} / \sum_{j \in Z_U} d_i^{e\prime}$
- ゾーン内に複数の停留所がある場合，ランダム選択される.
- ターミナル・停留所を選択した後にelementsが残っている場合ゼロを設定する.

ABCアルゴリズムによる解の生成・修復

(2) 修復手順

(2)-1: ゾーンカバーのチェック

- 各ゾーンが1つ以上の路線によってカバーされていることを保証（初期条件下ではカバーされていない可能性がある）.
- セントロイドmに対して路線が提供されていない場合，mと接続する停留所ノードiを停留所上限制約に抵触しないルートに追加する.
- もしそのようなルートが残っていない場合，2つ以上の路線が提供されているゾーンをランダム選択し，ゾーン内の停留所ノードの1つを停留所ノードiに置き換える.

(2)-2: 停留所配列の最適化

- ABCアルゴリズムによって生成されたルートに対し，降下法で停留所配列を改善.
- 改善の目的は各ルートの旅行時間の最小化.

ABCアルゴリズムによる解の生成・修復

■停留所配列の最適化

```
For each route in the ABC solution
Set i′ = 1
  While i′ ⩽ the number of intermediate stops − 1
    j′ = i′ + 1
    While j′ ⩽ the number of intermediate stops, do
        Exchange the i′th and j′th stops
        Evaluate the trip time of the route
        If the trip time is reduced,
          then set j′ = number of intermediate stops + 1, i′ = 0
        else
          undo the exchange and j′ = j′ + 1
        endif
      endwhile
    i′ = i′ + 1
  endwhile
Next route in the ABC solution
```

→ ノードの選定→ノード入れ替え→旅行時間算出→ノード決定→ノード更新

ABCアルゴリズムによる解の生成・修復

(2)-3: 停留所の削除・追加

- 停留所入れ替えだけでは最大旅行時間制約（制約条件(12)）を違反する可能性があるため，ゾーンカバー制約を確保しつつ停留所ノードを削除してクリアさせる．
- ルートrからノードiを削除した際の上位問題の目的関数値の変化を捉える指標として average-direct-demand $\psi_r^{ie'}$（直行便に対する平均需要）を採用．

$$\psi_r^{ie'} = \frac{d_i^{e'}\,\delta_r^{ie'}}{\sum_{p\neq r}\delta_p^{ie'}+1} \quad \text{for} \quad r=1 \text{ to } R_{\max},\ i\in Z_U,\ e'\in V,$$

$\delta_p^{ie'}$: バイナリ変数（ルートpがノードi とターミナルe'両方に接続する場合1，そうでなければ0）

$\sum_{p\neq r}\delta_p^{ie'}$: ルート$r$からノード$i$ が除去された後でノードiとターミナルe'の間で運行されている直行便の便数．

$d_i^{e'}$: 下位問題より得られる需要量（ノードi からターミナルe' に向かう総流動量）

- $\psi_r^{ie'}$が小さい場合，削除による新規乗換発生量が少ないことを意味するため，ノードが除去されやすい．
- ノードの減少によるサービスエリアの縮小や乗換客の増加などの負の影響を保証するため，ノードの挿入を実施．最大旅行時間制約に達していない場合，ノードを最大限挿入する．

ABCアルゴリズムによる解の生成・修復

(3) 下限値の設定と適応度評価

- 各反復の解の評価の際に適応度関数を活用する.
- 適応度関数の定式化の際には下限値（最小乗換客数）を用いて計算負荷軽減を図る.

■下限値の設定

$$LB_g = \underbrace{\sum_{i \in Z_U \setminus V} \sum_{e' \in V} d_i^{e'}}_{\text{総需要}} - \underbrace{\sum_{i \in Z_U \setminus V} \sum_{e' \in V} \left(1 - NR_i^{e'}\right) d_i^{e'}}_{\text{乗換なし乗客数}},$$

(23) ネットワーク接続と需要の状況から上位問題の目的関数値（乗換客数）の最小値LB_gを設定

where

$$NR_i^{e'} = \prod_{r=1}^{R_{\max}} (1 - RT_{ie'r}) \quad \text{for} \quad i \in Z_U \setminus V,\ e' \in V,$$

(24) ノードiとターミナルeの接続ルートがなければ1，あれば1を返すダミー変数$NR_i^{e'}$を設定

$$RT_{iwr} = X_{iwr} + \sum_{j \in Z_U \setminus V, j \neq i, j \neq w} X_{ijr} RT_{jwr} \quad \text{for} \quad i \in Z_U \setminus V,\ w \in Z_U \setminus V,\ w \neq i,\ r = 1 \text{ to } R_{\max},$$

ルートrがノードiのすぐ後にノードwを通過するとき1，そうでなければ0を返すダミー変数X_{iwr}

(25) ルートrがノードiとノードwを通過していれば1，そうでなければ1を返すダミー変数RT_{iwr}を設定

ABCアルゴリズムによる解の生成・修復

■適応度関数

- 適応度関数は下限値LB_gとペナルティ項P_gにより設定される.
- ペナルティ項は最大旅行時間及び最大車両台数の制約条件からの逸脱により評価され，値が大きい場合，その解の選択可能性が低くなるように設定されている.

$$F_g = \frac{1}{LB_g + P_g}. \qquad (26)$$

$$P_g = \underbrace{\alpha \sum_{r=1}^{R_{\max}} \max(T_{r,g} - T_{\max}, 0)}_{\text{最大旅行時間に対するペナルティ}} + \underbrace{\beta \max\left(\sum_{r=1}^{R_{\max}} V_{r,g} - W, 0\right)}_{\text{最大車両台数に対するペナルティ}}, \qquad (27)$$

$V_{r,g}$：解gルートrの車両台数,

W：ネットワークにおける最大車両台数,

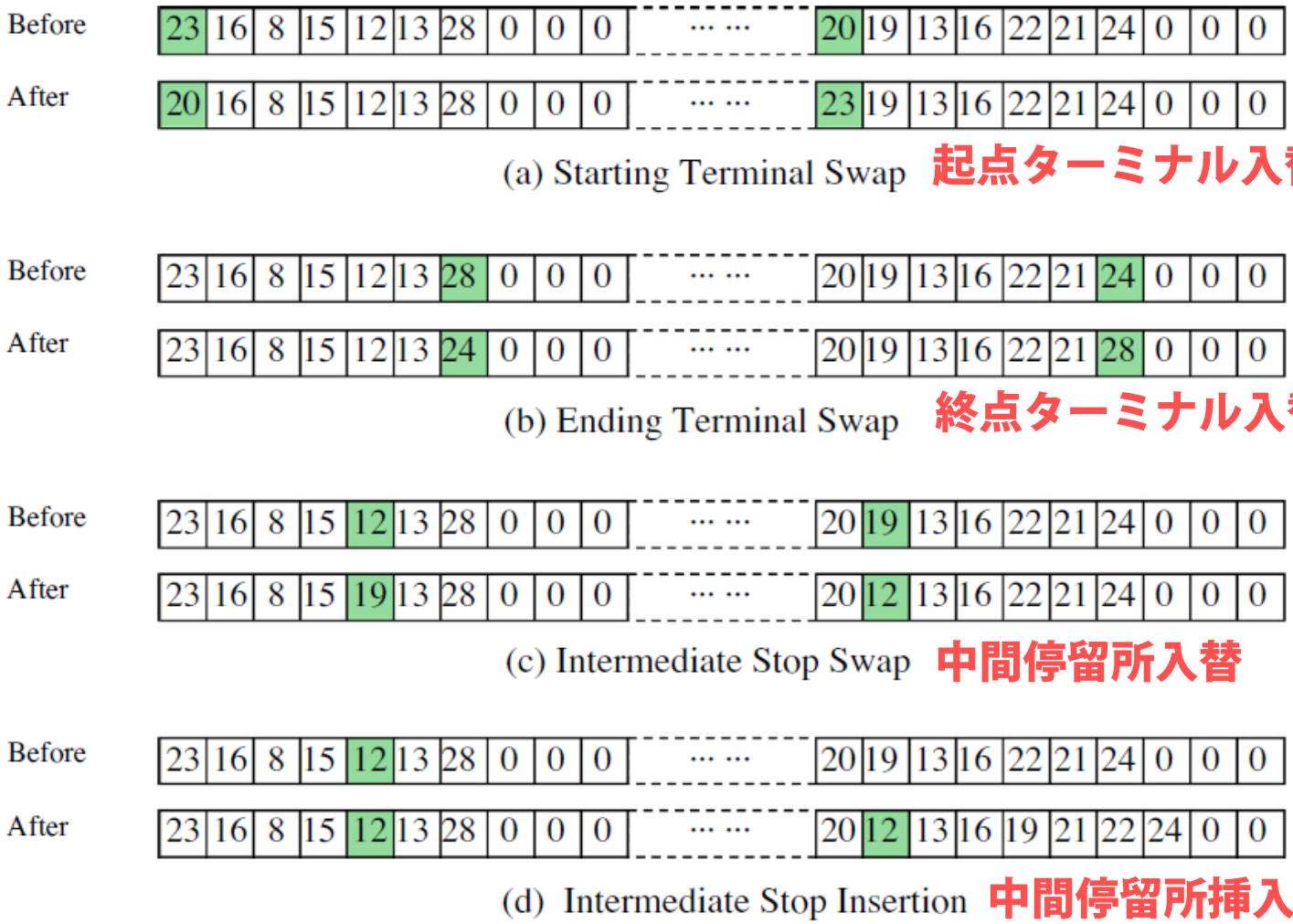
$T_{r,g}$：解gルートrの旅行時間,

T_{max}：Study Area内の最大旅行時間,

α, β：最大旅行時間と最大車両台数に係るパラメータ.

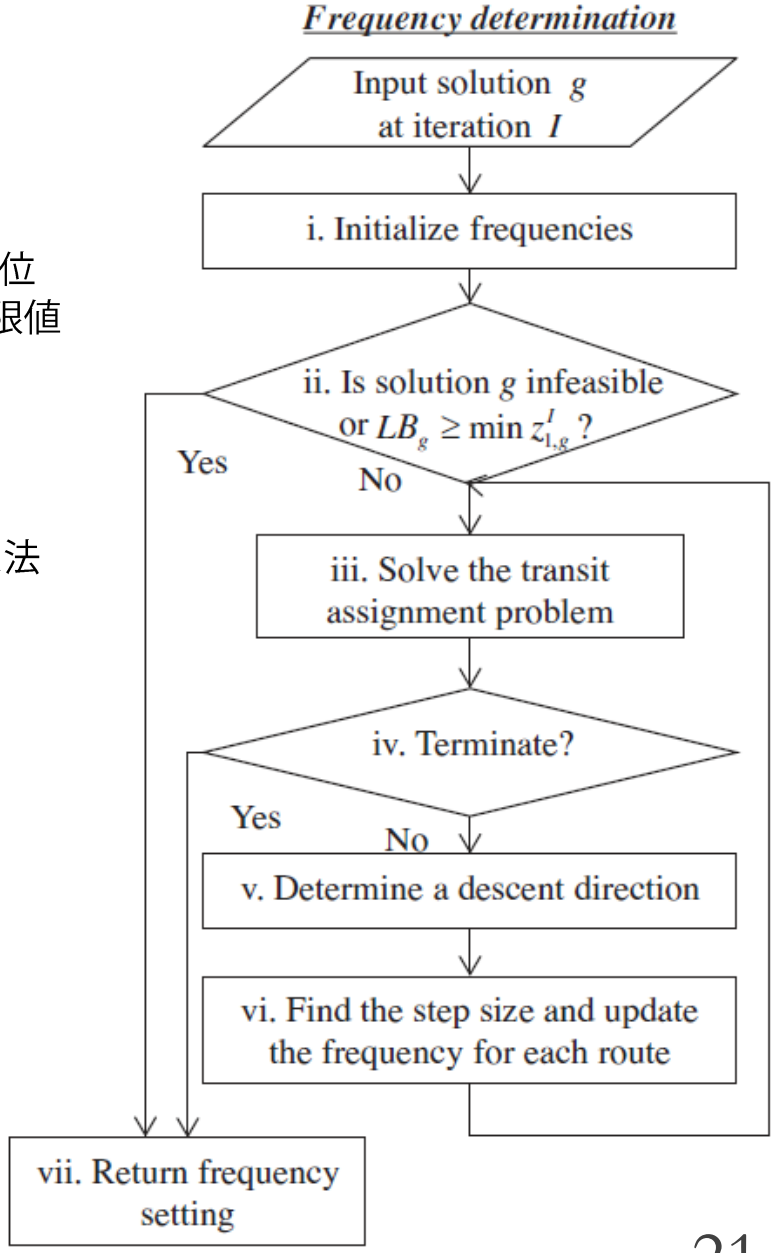
ルート設計における近傍探索

- 近傍探索フェーズではルート間での組み合わせにより4通りの探索方法が採用される.
- ここではノードの削除は行われない（最良解の探索に繋がらないため）.



便数決定アルゴリズムの流れ

- **Step i: 初期便数の設定**
  - $f_{r,g} = V_{r,g}/2T_{r,g}$ （$V_{r,g}$: 車両台数， $T_{r,g}$:旅行時間）
- **Step ii: 下限値のスクリーニング**
  - その時点でABCアルゴリズムによって得られている上位問題の最良解$z^I_{1,g}$と比較して，新しく探索する解の下限値(最適解)LB_gの方が大きい場合，解の探索をストップ
  - 実現不可能ルートの場合は解の探索をストップ
- **Step iii: 乗客配分問題（※下位問題）**
  - 与えられた便数とルート条件のもとでシンプレックス法で乗客配分問題を解く
- **Step iv: 終了判定**
- **Step v: 降下方向の決定**
- **Step vi: ステップサイズと便数の更新**
- **Step vii: 便数設定**



便数決定アルゴリズムの流れ

Step iv: 終了判定

- 終了基準は以下の通り.

$$\text{Criterion (1) } z_{1,g}^{k} - LB_g \leqslant \varepsilon_1 \text{ and}$$
$$\text{Criterion (2) } z_{1,g}^{k+1} - z_{1,g}^{k} \leqslant \varepsilon_2,$$

$\varepsilon_1\ and\ \varepsilon_2$：最大許容誤差（事前に設定），

$z_{1,g}^{k}$：反復k回後の上位問題目的関数の最良値.

Criterion (1)

- 便数が最適もしくはほぼ最適に至った場合.
- $z_{1,g}^{k} = LB_g$の場合は最適解であり，両者の差分が非常に小さい場合はほぼ最適であるとみなせる.

Criterion (2)

- 反復k回後と$k+1$回後の便数が等しいかほぼ等しい場合，最新の解を最適解とする.

便数決定アルゴリズムの流れ

Step v: 降下方向の決定

- 上位問題の運行便数の降下方向を決定.
- 乗換ペナルティが十分に大きい場合，下位問題の降下方向と上位問題の降下方向が一致するため，ここでは下位問題をもとに降下方向の決定方法を整理する.

■下位問題のベクトル標記　※(17)～(22)に対応

$$\min_{\mathbf{v},\mathbf{w}} : z_2 = h_1(\mathbf{v}(\mathbf{f})) + h_2(\mathbf{w}(\mathbf{f})), \quad \textbf{(29)}$$

Subject to:

$$\mathbf{g}_1(\mathbf{v}(\mathbf{f})) - \mathbf{k}(\mathbf{f}) \cdot \mathbf{m}(\mathbf{w}(\mathbf{f})) \leqslant \mathbf{0}, \quad \textbf{(30)}$$

$$\mathbf{g}_2(\mathbf{v}(\mathbf{f})) - \mathbf{d} = \mathbf{0}, \quad \textbf{(31)}$$

$$\mathbf{g}_3(\mathbf{v}(\mathbf{f})) - \mathbf{c}(\mathbf{f}) \leqslant \mathbf{0}, \quad \textbf{(32)}$$

$$\mathbf{v}(\mathbf{f}) \geqslant \mathbf{0}, \quad \textbf{(33)}$$

$$\mathbf{w}(\mathbf{f}) \geqslant \mathbf{0}, \quad \textbf{(34)}$$

where $\mathbf{v}(\mathbf{f})$ and $\mathbf{w}(\mathbf{f})$ represent the vectors $[v_a^e]$ and $[\omega_i^e]$, respectively, which are functions of frequencies. $h_1(\mathbf{v}(\mathbf{f})) = \sum_a \sum_e c_a v_a^e$ and $h_2(\mathbf{w}(\mathbf{f})) = \sum_i \sum_e \omega_i^e$. For constraints (30)–(32), $\mathbf{g}_1(\mathbf{v}(\mathbf{f})) = [v_a^e]$; $\mathbf{k}(\mathbf{f}) = [f_a]$; $\mathbf{m}(\mathbf{w}(\mathbf{f})) = [\omega_i^e]$; $\mathbf{g}_2(\mathbf{v}(\mathbf{f})) = \left[\sum_{a \in A_i^+} v_a^e - \sum_{a \in A_i^-} v_a^e\right]$; $\mathbf{d} = [d_i^e]$; $\mathbf{g}_3(\mathbf{v}(\mathbf{f})) = \left[\sum_e v_a^e\right]$ and $\mathbf{c}(\mathbf{f}) = [f_a k_{\text{cap}}]$. The dimensions of these matrices are not fixed, but vary with the solutions of the upper-level problem.

便数決定アルゴリズムの流れ

- 乗換ペナルティMが十分に大きい場合，下位問題の降下方向と上位問題の降下方向が一致すする

$$z_2 = \mathbf{C}^T\mathbf{x} + M\sum_{t\in T^R}\sum_{e\in G_d} v_t^e.$$

$\sum_{t\in T^R}\sum_{e\in G_d} v_t^e$: 全乗換リンクのトータルフロー，

x: 下位問題のその他の決定変数
M，C：パラメータ

・現行の便数は無限大よりも小さい．各路線の待ち時間は便数が無限大に近づくとゼロに近づく．つまり，下位問題の目的関数の中での待ち時間項を最小化すればよい．
・よって，-∇fLは必ずネガティブ．
・解の探索方向だけ探すのであれば，制約条件は考えず目的関数のみ考慮すればよい．このとき，MがCに対して十分大きい場合，Mの最小化を考えればよい．

便数決定アルゴリズムの流れ

■Karush-Kuhn-Tackerの定理を適用

$$\begin{pmatrix} \nabla_{\mathbf{v}} h_1(\mathbf{v}^*(\mathbf{f})) \\ \nabla_{\mathbf{w}} h_2(\mathbf{w}^*(\mathbf{f})) \end{pmatrix} + \pi^{\mathrm{T}} \begin{pmatrix} \nabla_{\mathbf{v}} \mathbf{g}_1(\mathbf{v}^*(\mathbf{f})) \\ (-\mathbf{k}(\mathbf{f})) \cdot \nabla_{\mathbf{w}} \mathbf{m}(\mathbf{w}^*(\mathbf{f})) \end{pmatrix} + \boldsymbol{\varphi}^{\mathrm{T}} \begin{pmatrix} \nabla_{\mathbf{v}} \mathbf{g}_2(\mathbf{v}^*(\mathbf{f})) \\ \mathbf{0} \end{pmatrix} + \boldsymbol{\mu}^{\mathrm{T}} \begin{pmatrix} \nabla_{\mathbf{v}} \mathbf{g}_3(\mathbf{v}^*(\mathbf{f})) \\ \mathbf{0} \end{pmatrix} = \mathbf{0}, \quad (35)$$

$$\begin{pmatrix} \pi \\ \boldsymbol{\mu} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{g}_1(\mathbf{v}^*(\mathbf{f})) - \mathbf{k}(\mathbf{f}) \cdot \mathbf{m}(\mathbf{w}^*(\mathbf{f})) \\ \mathbf{g}_3(\mathbf{v}^*(\mathbf{f})) - \mathbf{c}(\mathbf{f}) \end{pmatrix} = \mathbf{0}, \quad (36)$$

$$\begin{pmatrix} \pi \\ \boldsymbol{\mu} \end{pmatrix} \geqslant \mathbf{0}, \quad (37)$$

$$\begin{pmatrix} \mathbf{v}^*(\mathbf{f}) \\ \mathbf{w}^*(\mathbf{f}) \end{pmatrix} \geqslant \mathbf{0}, \quad (38)$$

■ラグランジュ関数をfで偏微分

$$\begin{aligned} \nabla_{\mathbf{f}} L = {} & \frac{\partial \mathbf{v}^*}{\partial \mathbf{f}} \cdot \nabla_{\mathbf{v}} \mathbf{h}_1(\mathbf{v}^*(\mathbf{f})) + \frac{\partial \mathbf{w}^*}{\partial \mathbf{f}} \cdot \nabla_{\mathbf{w}} \mathbf{h}_2(\mathbf{w}^*(\mathbf{f})) \\ & + \pi^{\mathrm{T}} \left(\frac{\partial \mathbf{v}^*}{\partial \mathbf{f}} \cdot \nabla_{\mathbf{v}} \mathbf{g}_1(\mathbf{v}^*(\mathbf{f})) - \frac{\partial \mathbf{k}(\mathbf{f})}{\partial \mathbf{f}} \cdot \mathbf{m}(\mathbf{w}^*(\mathbf{f})) - \frac{\partial \mathbf{w}^*}{\partial \mathbf{f}} (-\mathbf{k}(\mathbf{f})) \cdot \nabla_{\mathbf{w}} \mathbf{m}(\mathbf{w}^*(\mathbf{f})) \right) + \boldsymbol{\varphi}^{\mathrm{T}} \cdot \frac{\partial \mathbf{v}^*}{\partial \mathbf{f}} \cdot \nabla_{\mathbf{v}} \mathbf{g}_2(\mathbf{v}^*(\mathbf{f})) \\ & + \boldsymbol{\mu}^{\mathrm{T}} \left(\frac{\partial \mathbf{v}^*}{\partial \mathbf{f}} \cdot \nabla_{\mathbf{v}} \mathbf{g}_3(\mathbf{v}^*(\mathbf{f})) - \frac{\partial \mathbf{c}(\mathbf{f})}{\partial \mathbf{f}} \right). \quad (39) \end{aligned}$$

$$\begin{aligned} \nabla_{\mathbf{f}} L = {} & \frac{\partial \mathbf{v}^*}{\partial \mathbf{f}} \left\{ \nabla_{\mathbf{v}} \mathbf{h}_1(\mathbf{v}^*(\mathbf{f})) + \pi^{\mathrm{T}} \nabla_{\mathbf{v}} \mathbf{g}_1(\mathbf{v}^*(\mathbf{f})) + \boldsymbol{\varphi}^{\mathrm{T}} \nabla_{\mathbf{v}} \mathbf{g}_2(\mathbf{v}^*(\mathbf{f})) + \boldsymbol{\mu}^{\mathrm{T}} \nabla_{\mathbf{v}} \mathbf{g}_3(\mathbf{v}^*(\mathbf{f})) \right\} \\ & + \frac{\partial \mathbf{w}^*}{\partial \mathbf{f}} \left\{ \nabla_{\mathbf{w}} \mathbf{h}_2(\mathbf{w}^*(\mathbf{f})) + \pi^{\mathrm{T}} \cdot (-\mathbf{k}(\mathbf{f})) \cdot \nabla_{\mathbf{w}} \mathbf{m}(\mathbf{w}^*(\mathbf{f})) \right\} - \pi^{\mathrm{T}} \frac{\partial \mathbf{k}(\mathbf{f})}{\partial \mathbf{f}} \cdot \mathbf{m}(\mathbf{w}^*(\mathbf{f})) - \boldsymbol{\mu}^{\mathrm{T}} \frac{\partial \mathbf{c}(\mathbf{f})}{\partial \mathbf{f}}. \quad (40) \end{aligned}$$

便数決定アルゴリズムの流れ

■現在の最良解のラグランジュ関数の最急上昇方向

- (35)を(40)に代入することで算出．$-\nabla_{\mathbf{f}}L$はそのポイントにおける最急降下方向を意味する．
- なぜなら，$\nabla_{\mathbf{f}}L = \nabla_{\mathbf{f}}z_2\ at\ (\mathbf{v}^*(\mathbf{f}), \mathbf{v}^*(\mathbf{f}), \pi, \varphi, \mu), -\nabla_{\mathbf{f}}L = -\nabla_{\mathbf{f}}z_2\ at\ that\ point.$

$$\nabla_{\mathbf{f}}L = -\boldsymbol{\pi}^{\mathrm{T}}\frac{\partial \mathbf{k}(\mathbf{f})}{\partial \mathbf{f}} \cdot \mathbf{m}(\mathbf{w}^*(\mathbf{f})) - \boldsymbol{\mu}^{\mathrm{T}}\frac{\partial \mathbf{c}(\mathbf{f})}{\partial \mathbf{f}}. \quad (41)$$

便数決定アルゴリズムの流れ

Step vi: ステップサイズの決定と便数の更新

- 便数fを更新するためのステップサイズは以下の整数線形計画問題により決定.

■目的関数

$$\min_{\Delta \mathbf{v}} : z_3 = \sum_r \Delta f_r \cdot \frac{\partial L}{\partial f_r}, \qquad \textbf{(44)}$$

■制約条件

$$\Delta f_r = \frac{\Delta V_r}{2T_r}, \quad \text{for} \quad r = 1 \text{ to } R_{\max}, \quad \textbf{(45)ステップサイズは初期便数の設定方法に準拠}$$

$$\Delta f_r + f_r \geqslant f_{\min}, \quad \text{for} \quad r = 1 \text{ to } R_{\max}, \quad \textbf{(46) 最小頻度制約　※上位問題(10)と対応}$$

$$\sum_r (\Delta f_r + f_r) \cdot k_{\text{cap}} \cdot \delta_r^e \geqslant \sum_{m \in G_s} d_m^e, \quad \forall e \in G_d, \quad \textbf{(47) 容量制約　※上位問題(14)と対応}$$

$$\sum_r \Delta V_r = 0, \quad \textbf{(48) 車両台数制約}$$

ΔV_r：ルートrの車両台数の変化,
Δf_r：ルートrの便数のステップサイズの変化.

ケーススタディ

■小規模ネットワーク

- 仮想ネットワークを用いて本研究で構築したモデル・アルゴリズムの動作確認
- パラメータ設定の影響を確認.

■TSW（Tin Shui Wai, Hong Kong）ネットワーク

- 現実ネットワークへの適用可能性の確認.
- 適合度関数の影響，ABCアルゴリズムの設計条件の影響を確認.
- 非効率な路線体系により乗換が多く発生，バス稼働率低い．バス車両台数を拡大せずに乗換数が減らせるような再編を求める地元ニーズあり.

■Winnipeg（Canada）ネットワーク

- GAとの比較による計算速度向上効果の確認.

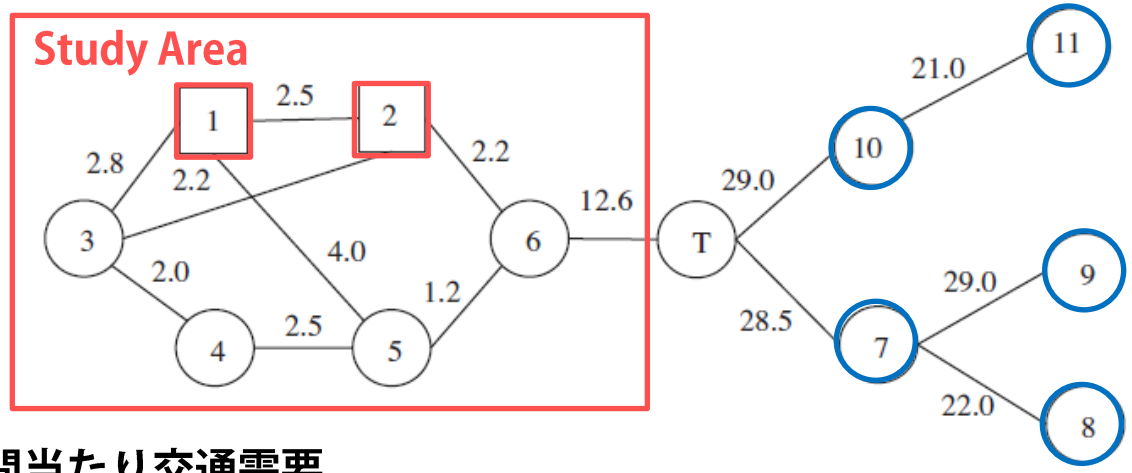
≪基本条件≫

- 乗換アークの旅行時間$M = 2000$; ABCアルゴリズムのコロニーサイズ$N_c = 100,$; 働き蜂数$N_e = 50,$; 見学蜂数$N_o = 50$; 働き蜂が偵察蜂に変化する閾値$limit = 50,$; 反復回数$I_{max} = 500$.
- ペナルティ項のパラメータ$\alpha, \beta = 10^9$; 頻度決定アルゴリズムの最大許容誤差$\varepsilon_1, \varepsilon_2 = 0.01$.
- ABCアルゴリズムはC++でコーディングしVisual Studio 2008に準拠．下位問題と整数線形計画問題はCPLEX12.4使用．いずれのケーススタディについても20回計算して平均値を採用.

小規模ネットワーク

■ネットワーク図

- 起点ターミナル：1,2，終点ターミナル：7-11，Study Area内ノード：1-6，セントロイド＝停留所ノードとする.
- 各ルートの最大旅行時間：23分，最大中間停留所数：3，ネットワーク内の総車両台数：60台，ルート最大数：5.



■ゾーン間1時間当たり交通需要

From/To	7	8	9	10	11	Total
1	192	148	102	94	149	685
2	100	74	78	56	102	410
3	87	77	71	46	113	394
4	96	63	49	34	85	327
5	33	24	19	15	34	125
6	19	14	14	9	23	79
Total	527	400	333	254	506	2020

小規模ネットワーク

■ベンチマーク

- 総当たり法により最適値を探索→最適乗換数411.
- ABCアルゴリズムにより最適値を探索→計算実施時の最適乗換数の平均・最小ともに411.
- ネットワークが単純な場合総当たりでも最適解の探索は可能.

Optimal solution 1			Optimal solution 2		
Stop sequence	Trip time (min)	Headway (min)	Stop sequence	Trip time (min)	Headway (min)
1, 3, 2, T, 8	22.8	13.3	1, 3, 2, T, 8	22.8	12.2
1, 3, 2, T, 9	22.8	12.4	1, 3, 2, T, 9	22.8	11.5
1, 3, 2, T, 11	22.8	13.2	1, 3, 2, T, 11	22.8	12.1
2, 5, 6, T, 10	20.2	12.3	2, 6, 5, T, 10	20.2	7.6
2, 4, T, 7	22.0	5.9	2, 4, T, 7	22.0	11.2

■下限スクリーニングの影響

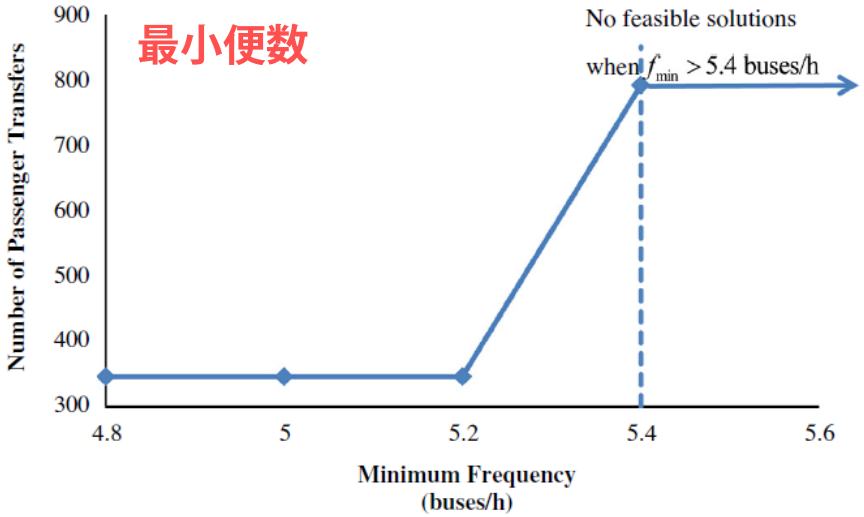
- 下限スクリーニングにより最適解候補のみ最適化計算に使用されたこと（解の絞込み）と便数探索の終了基準が厳格化されたこと（反復回数の削減）によって計算時間が大幅に削減.
- 総当たり法と比較してABCアルゴリズムは大幅に計算時間削減.

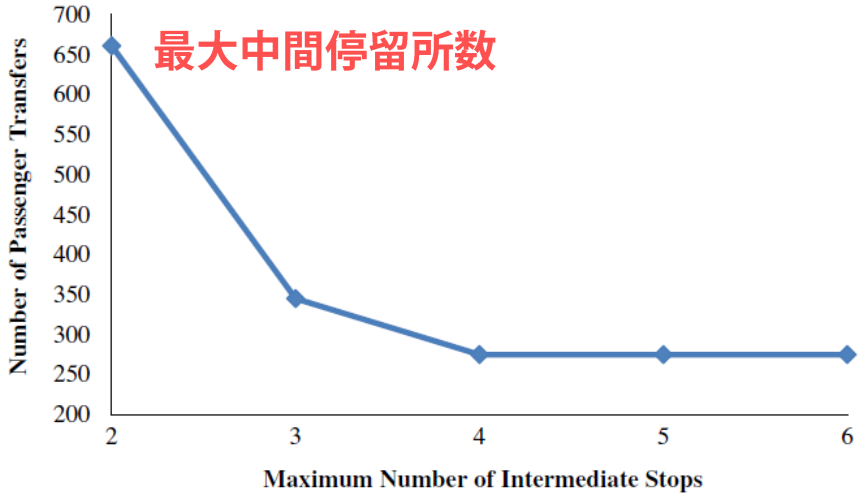
	Brute force method		Hybrid ABC algorithm	
With lower bound screening	No	Yes	No	Yes
Average computation time (seconds)	8292.55	14.16	1003.80	0.20

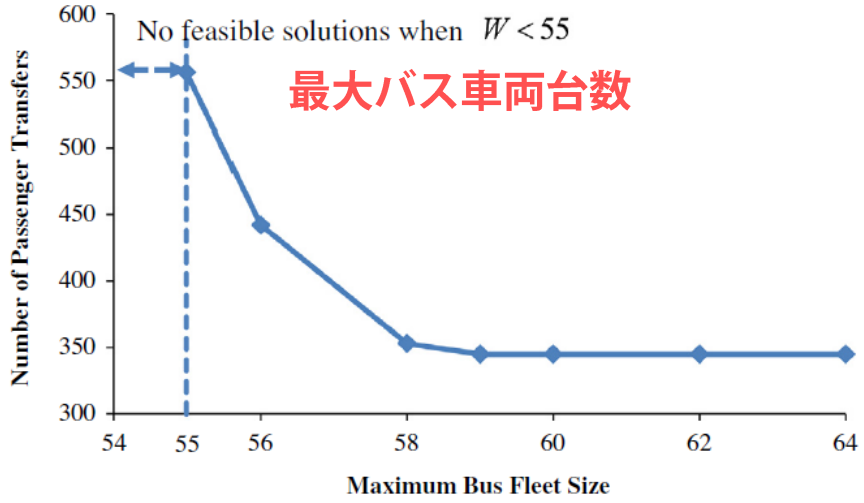
小規模ネットワーク

■パラメータ設定の影響

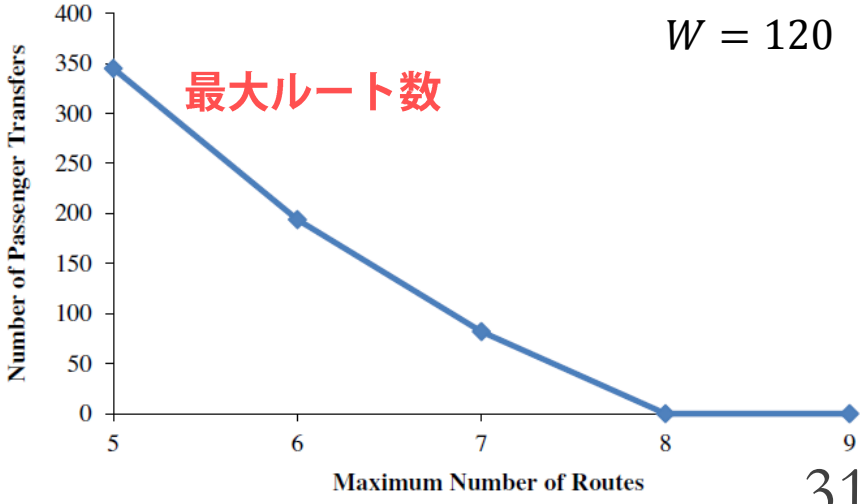
$f_{min} = 4.8, W = 60, S_{max} = 3, R_{max} = 5, and\ T_{max} = 26.$



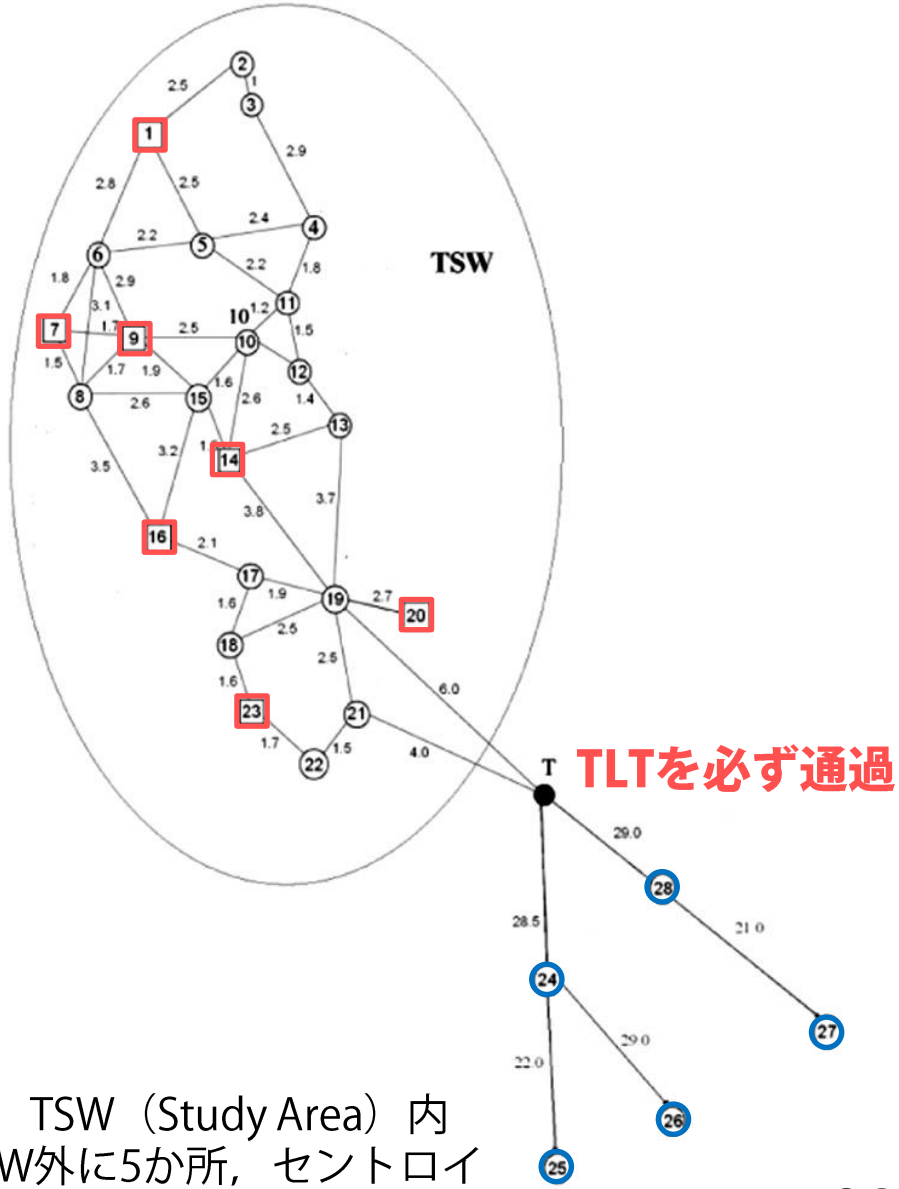




$W = 120$



TSWネットワーク



- □が起点ターミナル，○が停留所ノード，TSW（Study Area）内を23ゾーンに分割，終点ターミナルはTSW外に5か所，セントロイド＝停留所ノードとする.

TSWネットワーク

From/To	24	25	26	27	28	Total
1	192	148	102	94	149	685
2	54	39	38	22	54	207
3	47	40	38	27	55	207
4	33	22	21	14	30	120
5	100	74	78	56	102	410
6	87	77	71	46	113	394
7	113	76	71	46	103	409
8	100	76	71	47	117	411
9	96	63	49	34	85	327
10	33	24	19	15	34	125
11	19	14	14	9	23	79
12	156	134	114	69	165	638
13	177	105	90	78	143	593
14	63	48	36	29	59	235
15	102	81	63	39	93	378
16	253	170	150	127	213	913
17	28	20	20	14	27	109
18	76	63	58	38	71	306
19	34	25	22	14	30	125
20	59	39	30	26	49	203
21	36	23	22	15	28	124
22	33	25	20	16	28	122
23	206	184	147	96	209	842
Total	2097	1570	1344	971	1980	7962

TSWネットワーク

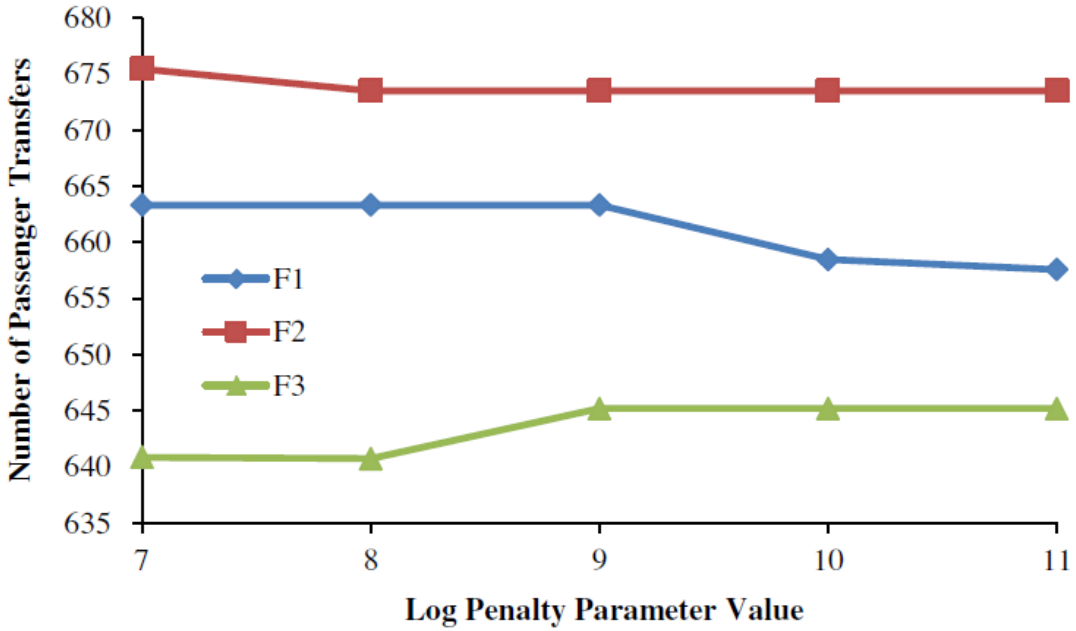
■適応度関数とペナルティパラメータの影響

- F3の関数形（本研究のデフォルトの式形）で最良解.
- ペナルティパラメータについては$\alpha, \beta = 10^8$のとき最良解（$\alpha = \beta$で仮定）.
- パラメータの違いよりも関数形の違いの方が支配的.

$$F1_g = 10^{15} - LB_g - P_g,$$

$$F2_g = C_I - LB_g - P_g, \text{ and}$$

$$F3_g = \frac{1}{LB_g + P_g}.$$



$$P_g = \alpha \sum_{r=1}^{R_{\max}} \max(T_{r,g} - T_{\max}, 0) + \beta \max\left(\sum_{r=1}^{R_{\max}} V_{r,g} - W, 0\right),$$

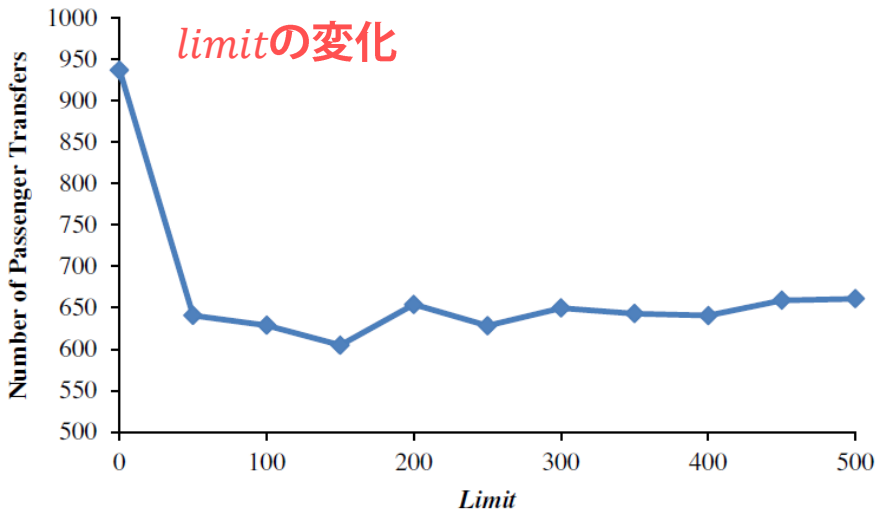
TSWネットワーク

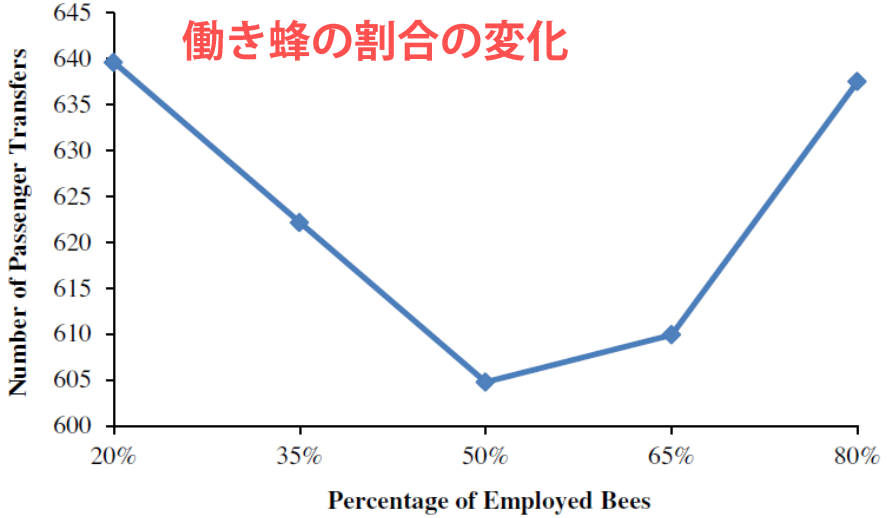
■働き蜂→偵察蜂変化の閾値*limit*の影響

- $limit = 0$の場合全ての解は反復ごとに破棄され，都度ランダム生成される．
- $limit = 150$までは最良解が改善，それ以降は悪化．
- *limit*が小さいと解空間の探索が十分にできず，大きすぎると無駄な探索が増える．

■コロニー構成の影響

- コロニーサイズと反復回数は固定（働き蜂が増えると見学蜂が減る）．
- 両者が50%ずつの時に最良解．
- 働き蜂が多いと探索される餌場が限定されてしまうし，少ないと近傍解の探索が限定されてしまう．





TSWネットワーク

■ノード挿入・削除戦略の違いによる影響

- ノードの挿入と削除の際の指標を以下の3パターンで比較.
  - S1: average-direct-demandにより設定（デフォルト）
  - S2: 移動費用変化により設定
  - S3: total-direct-demandにより設定
- 需要量はnormal demand（100%）とlow demand(50%)で比較.
- S1が優位．特に需要が大きい場合に顕著.
- 費用や総需要の変化の場合，ノードにおける需要の拡大や距離の短縮は便数の拡大につながり，他のノードに乗り換える誘導になる.

Table 3
Comparison of the different insertion and deletion strategies.

	S1	S2	S3
50% of the Demand (low demand)	311.62	646.05 (+107.3%)	602.62 (+93.4%)
100% of the Demand (normal demand)	604.76	1313.99 (+117.3%)	1207.59 (+99.7%)

TSWネットワーク

■不確定需要下での影響

- 需要の日変動に対するアルゴリズムの眼鏡性を確認するため，1000通りの需要マトリクスを無秩序に生成（ノードmからターミナルeまでの需要は一様分布$[0.8d_m^e, 1.2d_m^e]$で設定）し計算，現行サービスの実態と比較することで評価.
- ABCアルゴリズム活用により平均乗換客数が大幅減，サービス容量の不足によるunserved demandはゼロ.
- ABCアルゴリズムによる最良解は現行と比較して停留所数が減少，かつ運行間隔が平準化（Std. 2.9 min → 2.4minに減少）

Table 4
Comparison of the solutions under random demand.

	Average No. of transfers	Std. of No. of transfers	Average unserved demand	Std. of unserved demand
Hybrid ABC algorithm	355.491	22.38	0.00	0.00
Current	1563.31	35.95	1350.85	56.18

TSWネットワーク

■不確定需要下での影響

Table 5
Existing route structures and headways.

現行

Routes	Stop sequence	Number of buses	Headway (min)
1	20, 19, T, 25	12	9.8
2	16, 17, 18, 23, 22, 21, T, 25	17	8.1
3	16, 17, 18, 23, 22, 21, T, 24	19	5.1
4	1, 6, 9, 10, 12, 13, 19, 21,T, 25	19	8.5
5	1, 6, 5, 4, 11,12, 13, 19, T, 24	23	5.3
6	14, 15, 8, 9, 10, 12, 13, 19, T, 24	11	11.1
7	1, 6, 8, 16, 17, 18, 23, 22,21, T, 28	30	4.1
8	14, 13, 12, 10, 8, 16, 17, 18, 23, 22, T, 26	18	11.0
9	7, 6, 1, 2, 3, 4, 11, 12, 13, 19, T, 24	11	12.3
10	9, 10, 11, 5, 6, 8, 16, 17, 18, 23, 22, T, 27	16	11.3

Table 6
Best solution obtained by the proposed ABC algorithm.

ABCアルゴリズムによる最適解

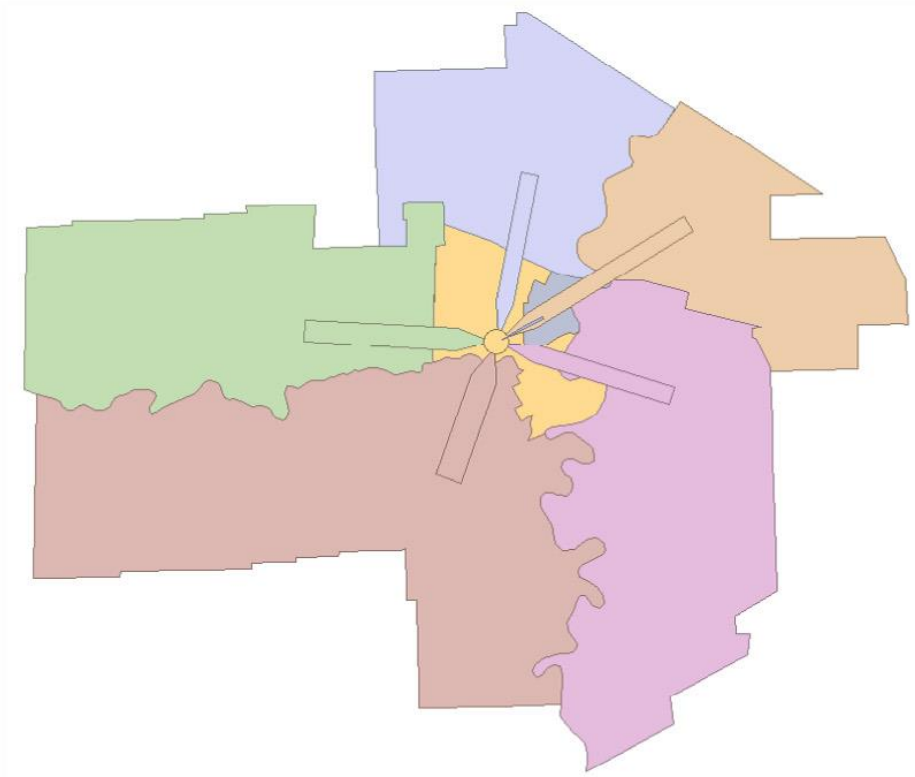
Routes	Stop sequence	Number of buses	Headway (min)
1	14, 13, T, 27	11	11.6
2	16, 1, 11, 13, 21, T, 24	23	5.3
3	20, 23, 18, 16, 15, 14, T, 26	20	9.2
4	7, 5, 1, 6, 9, 8, T, 25	14	12.1
5	9, 5, 6, 8, 16, 17, 22, T, 26	15	12.2
6	7, 1, 2, 3, 4, 12, 13,T, 26	15	12.3
7	16, 8, 18, 20, 22, 23, 13, 12, T, 28	25	6.7
8	23, 18, 16, 15, 10, 12, 13, 19, T, 25	19	8.9
9	1, 2, 3, 4, 5, 7, 9, 14, 19, T, 28	14	9.8
10	16, 23, 6, 10, 11, 15, 17, 21, 19, T, 28	20	8.5

Winnipegネットワーク

■ネットワーク図

- ゾーン数154，ノード数1,067，リンク数2,995，ゾーン数7，路線数30.
- 各路線の最大停留所数15，旅行時間45分.
- 最大車両数1200，バス1台あたりの容量60.

(a) Winnipeg Network



(b) Districts of Winnipeg Network

Winnipegネットワーク

■GAとの比較

- 公正な比較のためGAの解の表現，初期化手順，便数決定アルゴリズムについてはHybrid ABCアルゴリズムと揃えた.
- 人口規模＝コロニーサイズとして設定.
- GAは解の生成の際に交叉と突然変異を採用．ただし今回適用するGAは通常のGAと解の表現方法が異なるため，交叉についてはSzeto and Wu (2010)が提案した手法，突然変異については本研究で用いた近傍探索を採用.
- 各アルゴリズムとも20回計算．計算時間は300秒で設定.
- 平均値はHybrid ABCの方が優れており，標準偏差も小さく結果が安定（平均値の差はt値=2.093で5%有意）.

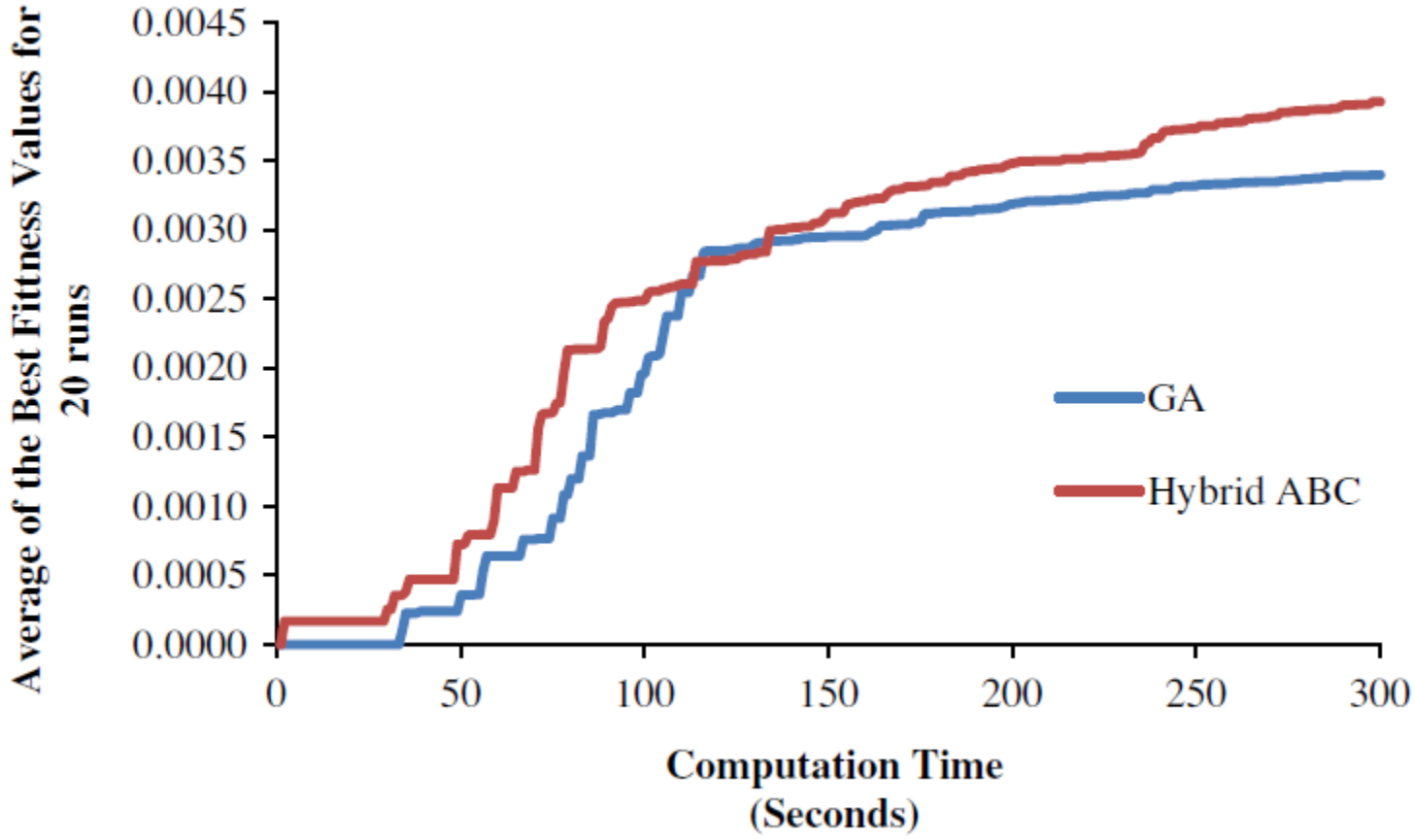
Table 7
Comparison of the computation performance of the GA and hybrid ABC algorithm.

Number of passenger transfers	GA	Hybrid ABC
Average	300.4	252.6 (−15.9%)
Standard deviation	47.6	29.0 (−39.0%)
Minimum	246.0	216.0 (−12.2%)

Winnipegネットワーク

■GAとの比較

- 計算時間ごとの適合度関数の値（20回の平均値）はいずれのプロセスでもHybrid ABCの方が優れており，同じ計算時間でもGAより優れていることを確認.



Conclusions

- 公共によるオペレーションと合理的な行動を仮定すれば現行モデルで十分だが，実際には政府や交通事業者等のステークホルダーの意図も考慮した多目的モデルへの拡張が必要.
- 意思決定において運賃は非常に重要な要素であり，運賃構造をルート選択に組み込むことの重要性.
- 線形計画法で解くために乗客配分の際の旅行時間は固定したが，実際には駅での乗降人数が増えることで待ち時間や乗換時間は増大．これらを考慮することはできるが，アルゴリズムの工夫が必要.

御清聴ありがとうございました

Notations

Sets

Z_U	a set of nodes in the upper-level network, excluding the depot
G_s	a set of centroids within the study area
H_m	a set of candidate stops connecting to centroid m
U	a set of starting bus terminals inside the study area
V	a set of ending terminals outside the study area
G_d	a set of centroids/destinations outside the study area
C	a set of bus terminals and candidate stops within the study area
Z_L	a set of nodes (including centroids) in the lower-level network
T^R	a set of transfer links or arcs in the lower-level network
A	a set of transit links in the lower-level network
A_i^+	a set of transit links coming out from node i; and
A_i^-	a set of transit links going into node i

Indices

i, j, m	indices of nodes
e	the index of a centroid/destination outside the study area
e'	the index of an ending bus terminal outside the study area; and
r	the route index

Parameters

c_{ij}	the in-vehicle travel time on the shortest path between nodes i and j
c_a	the in-vehicle travel time on link a
s_t	the average time for stopping at a node
d_m^e	the travel demand from node m to centroid e
W	the maximum bus fleet size allowed for the network
k_{cap}	the capacity of a bus
R_{max}	the maximum number of routes in the bus network
f_{min}	the minimum frequency of a route
S_{max}	the maximum number of stops (including the bus terminal) within the study area on a route
T_{max}	the maximum route travel time within the study area; and
p	a very large value used in the sub-tour elimination constraint

Notations

Decision Variables

Lower-level decisions

v_t^e	the number of passenger transfers on transit link t to destination e
v_a^e	the flow on link a to destination e
ω_i^e	the total waiting time at node i for all flows to destination e
v	$[v_a^e]$
w	$[\omega_i^e]$

Upper-level decisions

q_{ir}	the node potential at node i, which is needed in the sub-tour elimination constraint for bus route r
X_{ijr}	1 if route r (r = 1 to R_{max}) passes through node j immediately after node i, and 0 otherwise
X_{0jr}	1 if route r starts at node j, and 0 otherwise
X_{i0r}	1 if route r ends at node i, and 0 otherwise
X_{00r}	1 if route r is not available, and 0 otherwise
f_r	the frequency of route r
X	$[X_{ijr}]$; and
f	$[f_r]$

Functions of decision variables

T_r	the trip time of route r from the starting terminal to the ending terminal
δ_r^e	1 if route r connects the terminal that links to centroid e, and 0 otherwise
f_a	the frequency of link a; and
$d_i^{e'}$	the travel demand from node i to bus terminal e'

便数決定アルゴリズムの流れ

Step vi: ステップサイズの決定と便数の更新

- 便数fを更新するためのステップサイズを決定．ラグランジュ関数を便数f_rで偏微分する．

$$\frac{\partial L}{\partial f_r} = \underbrace{-\sum_i \sum_a \sum_e \pi_{ia}^e \cdot \frac{\partial f_a}{\partial f_r} \cdot \omega_i^{e^*}}_{\text{ノード交通流の分布に関する制約条件の双対解}} \underbrace{- \sum_a \mu_a \frac{\partial [f_a k_{\text{cap}}]}{\partial f_r}}_{\text{容量制約に関する制約条件の双対解}}, \qquad (43)$$

- リンクaがルートrの乗車アークである場合は$\frac{\partial f_a}{\partial f_r} = 1$，そうでなければ$\frac{\partial f_a}{\partial f_r} = 0$.
- リンクaがルートrのトラベルアークである場合は$\frac{\partial [f_a k_{cap}]}{\partial f_r} = k_{cap}$，そうでなければ $\frac{\partial [f_a k_{cap}]}{\partial f_r} = 0$.
- つまり，$\frac{\partial L}{\partial f_r}$はルート$r$の便数のみに影響を受けている．このような分離性質により，ルートごとに便数を設定したりステップサイズΔf_rを決定したりできる．