情報処理学会論文誌　数理モデル化と応用　Vol.7　No.1　94–105 (Mar. 2014)

予測交通量に基づくアントコロニー最適化法による時間依存TSPの解法と広域道路網への適用

落合 純一[1,a)]　狩野 均[2]

受付日 2013年8月26日，再受付日 2013年11月13日 / 2013年12月19日，
採録日 2014年1月17日

概要：本論文では，予測交通量を用いてアントコロニー最適化法（ACO）で時間依存 TSP（TDTSP）を解く手法を提案する．TDTSP とは，旅行時間が変化するタイプの TSP であり，ルーティング問題やスケジューリング問題など，いくつかの現実世界の問題をモデル化することができる．現実のルーティング問題を考えると，都市間の経路を求めてから旅行時間を計算する必要がある．旅行時間の変化間隔で良い解を求めるためには，探索の高速化が必要となるため，ACO のフェロモンの初期値に偏りを与えることで，探索領域の削減を行った．TSP のベンチマーク問題から TDTSP を作成した実験と，現実の道路網と交通量データを用いた実験の結果，提案手法は解の精度を落とすことなく収束が早まっていることを確認した．

キーワード：アントコロニー最適化法，巡回セールスマン問題，動的環境

Solving Time-dependent Traveling Salesman Problems Using Ant Colony Optimization Based on Predicted Traffic and Its Application to Wide Area Road Network

Junichi Ochiai[1,a)]　Hitoshi Kanoh[2]

Received: August 26, 2013, Revised: November 13, 2013/December 19, 2013,
Accepted: January 17, 2014

Abstract: In this paper, we propose an ant colony optimization based on the predicted traffic for time-dependent traveling salesman problems (TDTSP). TDTSP is a TSP where travel times between cities changes with time and used as models for some real world problems. In the case of real road networks, the travel time changing is associated with paths between cities and needs to be calculated by a shortest path algorithm. A faster algorithm is required to find the best solution every changing of travel times. The proposed method gives deviations from the initial pheromone trails in order to reduce a search space. Experimental results using benchmark problems and real world data suggested that the proposed method has a faster search rate than the conventional method.

Keywords: ant colony optimization, traveling salesman problem, dynamic environment

1. はじめに

アントコロニー最適化法（ACO）は，蟻の採餌行動をモデル化したメタヒューリスティクスであり，様々な組合せ最適化問題に適用されている [1], [2], [3], [4]．ACO では，探索領域の情報をフェロモンと見なし，複数の蟻がフェロモンに基づいて解を作成する．そして，作成された解の情報がフェロモンにフィードバックされ，更新されたフェロモンを利用して蟻が解を作成することで探索が進む．よっ

[1] 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[2] 筑波大学システム情報系情報工学域
Division of Information Engineering, Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[a)] ochiai@kslab.cs.tsukuba.ac.jp

て，フェロモンの扱い方は ACO の性能に大きく影響を与える [1].

現在までに ACO の性能向上に関する様々な論文が発表されており，多くのものは目的関数に時間要素が含まれていない静的な問題を対象としている．しかし，現実問題への適用を考える場合，目的関数に時間要素が含まれる動的な問題を解かなければならないことが多々ある．そこで，本論文では時間依存巡回セールスマン問題（TDTSP）を対象とする．TDTSP とは都市間の移動コストが変化するタイプの巡回セールスマン問題（TSP）であり，ルーティング問題やスケジューリング問題など，いくつかの現実世界の問題をモデル化することができる [5]．TDTSP を解く一般的な方法は，移動コストが変化するごとに再探索を行うものである [6], [7], [8]．しかし，この方法で見つけた解は最良解とは限らない．

本論文では，TDTSP の移動コストを旅行時間と考え，予測交通量 [9] を用いて ACO で TDTSP を解く手法を提案する．提案手法のベースとなる ACO には，安定して精度の良い解を求めることができる MAX-MIN Ant System (MMAS) [10] を用いた．さらに，MMAS のフェロモンの初期値に偏りを与えることで探索領域を削減し，探索の収束を早めることを図った．この理由は，MMAS には他の ACO と比較して収束が遅いという欠点があり [1]，TDTSP の旅行時間の変化間隔で精度の良い解を求めるためである．フェロモンの初期状態に偏りを与える手法として，Ant Colony System（ACS）[11] をベースとしている手法 [12], [13] と，MMAS をベースとしている手法 [14], [15] がある．提案手法は，MMAS をベースとしている手法 [14], [15] を TDTSP 向けに改良したものである．また，ACS をベースとしている手法 [12], [13] を MMAS に適用し，フェロモンの初期化に関する評価を行った．

TDTSP を広域道路網へ適用する場合，旅行時間の変化ごとに都市間の経路を計算する必要がある．時間依存問題を広域道路網に適用した研究 [16], [17] では，解の探索前に都市間の経路を求めており，その計算時間については言及されていない．本論文では，解の探索と同時に，必要に応じてダイクストラ法により都市間の経路を求めることで，経路探索の計算時間も含めて提案手法の評価を行う．

以下では，まず研究分野の概要として，TDTSP，ACO，ACO の高速化，広域道路網への適用について述べる．次に提案手法について述べる．最後に評価実験として，TSP のベンチマークを用いた実験，現実の道路網と交通量データを用いた実験について述べ，提案手法の有効性を示す．

2. 研究分野の概要

2.1 時間依存巡回セールスマン問題（TDTSP）

巡回セールスマン問題（TSP）は完全グラフ $G = (N, A)$ として表現できる [18]．N は頂点集合，A は辺集合であり，頂点数 $n = |N|$ となる．TSP では頂点を都市と呼び，辺には移動コストが与えられている．TSP の目的は，総移動コストが最小となるハミルトン閉路を求めることである．TSP のベンチマークとして TSPLIB [19] が一般的に用いられている．TSPLIB で公開されている問題は，都市の番号と座標，都市間の距離の計算方法が定義されており，距離を移動コストとして用いる．

TDTSP は，都市が解に現れる位置によって次の都市への移動コストが変化するタイプの TSP であり，最初に出発する都市が固定されているハミルトン閉路の中で，総移動コストが最小のものを求める問題である [5]．現実のルーティング問題を考えたとき，日本では道路交通情報通信システム（VICS）により 5 分間隔で最新の交通情報を得ることができる．そこで本研究では，都市間の移動コストを旅行時間とし，時間間隔 Δt で旅行時間が変化するものとして，TSPLIB の問題から TDTSP を作成した．TDTSP の解探索の概念図を**図 1** に示す．時間 t における都市 i，j 間の旅行時間 $T_{ij}(t)$ は，1 つ前の都市 i，j 間の旅行時間 $T_{ij}(t - \Delta t)$ に基づいて計算する．旅行時間の計算式を式 (1) に示す．

$$T_{ij}(t) = \begin{cases} T_{ij}^{\min} & \text{if } 0 \le t < \Delta t \\ \max(T_{ij}^{\min}, T_{ij}(t-\Delta t) \times (1+C_f \times R)) & \text{otherwise} \end{cases} \tag{1}$$

$T_{ij}^{\min}$ は都市 i，j 間の旅行時間の最小値，C_f は旅行時間の変化の大きさを表す範囲 $[0, 1]$ のパラメータ，R は範囲 $[-1, 1]$ の一様乱数である．$T_{ij}^{\min}$ は TSPLIB で定義されている都市 i，j 間の距離 d_{ij} として TDTSP を作成した．旅行時間 $T_{ij}(t)$ は解の探索前に既知であるものと仮定し，解 s の総旅行時間 $T(s)$ の計算（式 (2)，(3)）に用いる．

$$T(s) = \sum_{i=1}^{n} T_{i,i+1}(t_i) \tag{2}$$

$$t_i = \begin{cases} 0 & \text{if } i = 1 \\ t_{i-1} + T_{i-1,i}(t_{i-1}) & \text{otherwise} \end{cases} \tag{3}$$

2.2 アントコロニー最適化法（ACO）

ACO とは，蟻の採餌行動をモデル化したメタヒューリスティクスであり，様々な組合せ最適化問題に適用されている [1], [2], [3], [4]．ACO のアルゴリズムを**図 2** に示す [1]．まず，フェロモンを均一に初期化する．その後，複数の蟻が解を作成し，作成された解に基づいてフェロモンの更新が行われ，終了条件を満たすまでこの 2 つの処理が繰り返される．これにより，探索が進むにつれて徐々にフェロモンに偏りが生じる．これらの処理を改良することで様々な ACO が提案されており [1]，性能が良い代表的な ACO として MAX-MIN Ant System（MMAS）[10]，Ant Colony

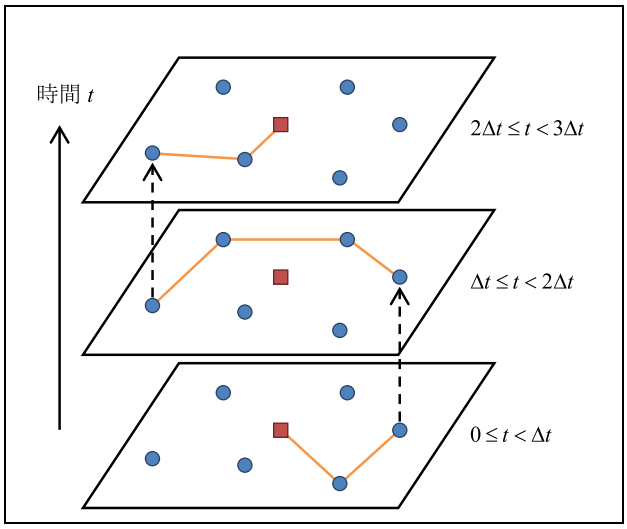

図 1　TDTSP の解探索の概念
Fig. 1　The idea of solution search in TDTSP.

```
procedure ACO ( )
  Set parameters
  Initialize pheromone trails
  while ( termination condition not met ) do
    Construct ant solutions
    Apply local search   // オプション
    Update pheromone trails
  end - while
end - procedure
```

図 2　ACO のアルゴリズム
Fig. 2　ACO algorithm.

System（ACS）[11] がある．

2.3　MAX-MIN Ant System（MMAS）

ここでは，通常の TSP を対象に MMAS [10] について述べる．MMAS は，まず初期解を作成してフェロモンの初期化を行う．フェロモンは都市間の重みとして値を持ち，初期解は Nearest Neighbor（NN）法 [20] により作成される．MMAS のフェロモンの初期値 τ_0 を式 (4) に示す．

$$\tau_0 = \frac{1}{\rho \times f(s_{\mathrm{NN}})} \tag{4}$$

s_{NN} は NN 法で作成した初期解，$f(s)$ は解 s の目的関数値，ρ はパラメータである．フェロモンの初期化では，すべての辺のフェロモンの値を τ_0 に初期化する．

フェロモンの初期化の後，終了条件を満たすまで蟻による解の探索とフェロモンの更新を行う．蟻 k が都市 i にいるとき，次に移動する都市として都市 j を選ぶ確率 p_{ij}^k を式 (5) に示す．

$$p_{ij}^k = \begin{cases} \dfrac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} & \text{if } j \in N^k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

N^k は蟻 k の未訪問都市集合，τ_{ij} は都市 i，j 間のフェロモンの値，α と β はパラメータである．η_{ij} はヒューリスティック値と呼ばれる問題固有の値であり，通常の TSP では距離の逆数を用いる．よって，都市 i，j 間のフェロモンの値が大きければ大きいほど，都市 i，j 間の距離が小さければ小さいほど，次に移動する都市として j が選ばれる確率が高くなる．

すべての蟻が解を作成した後，フェロモンの更新が行われる．フェロモンの更新式を式 (6)，(7) に示す．

$$\tau_{ij} \leftarrow (1-\rho) \times \tau_{ij} + \Delta\tau_{ij}^{\mathrm{best}} \tag{6}$$

$$\Delta\tau_{ij}^{\mathrm{best}} = \begin{cases} 1/f(s_{\mathrm{best}}) & \text{if } (i,j) \in s_{\mathrm{best}} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

s_{best} はフェロモンの更新に用いる解であり，問題によって適している解が異なるが，イテレーションごとの最良解 s_{ib} が一般的に用いられている．MMAS の特徴の 1 つにフェロモンの最大値 $\tau_{\max}$ と最小値 $\tau_{\min}$ がある．$\tau_{\max}$ と $\tau_{\min}$ はそれぞれ式 (8)，(9) から求める．

$$\tau_{\max} = \frac{1}{\rho \times f(s_{\mathrm{gb}})} \tag{8}$$

$$\tau_{\min} = \frac{1 - \sqrt[n]{p_{\mathrm{best}}}}{(n/2-1) \times \sqrt[n]{p_{\mathrm{best}}}} \times \tau_{\max} \tag{9}$$

s_{gb} は探索開始時からの最良解，p_{best} はパラメータである．よって，$\tau_{\max}$ と $\tau_{\min}$ は s_{gb} が更新されるごとに再計算される．p_{best} が表す意味は，最適解の辺のフェロモンの値が $\tau_{\max}$，それ以外の辺のフェロモンが $\tau_{\min}$ の状況で，ヒューリスティック値を無視した場合に蟻が最適解を作成する確率である．p_{best} の値は実験的に 0.05 が良い性能を示しており，この値が一般的に用いられている．

式 (4)，(8) より，MMAS の τ_0 は $\tau_{\max}$ に相当する．よって，フェロモンは大きい値で初期化され，探索が進むにつれて重要でない辺のフェロモンの値が小さくなる．

2.4　Ant Colony System（ACS）

ここでは，通常の TSP を対象に ACS [11] について述べる．まず，ACS のフェロモンの初期値を式 (10) に示す．

$$\tau_0 = \frac{1}{n \times f(s_{\mathrm{NN}})} \tag{10}$$

次に，蟻 k が都市 i にいるとき，次に移動する都市 j の求め方を式 (11) に示す．

$$j = \begin{cases} \arg\max_{l \in N^k} [\tau_{il}][\eta_{il}]^\beta & \text{if } q < q_0 \\ J & \text{otherwise} \end{cases} \tag{11}$$

q は範囲 $[0,1)$ の一様乱数，q_0 は範囲 $[0,1]$ のパラメータ，J は式 (5) で求めた都市である．

ACS では，一般的な ACO のパラメータである α は 1 として用いられる．確率 q_0 で選択確率が最大となる都市を

決定的に選択することから，MMAS と比べて探索の集中化が強められている．

最後に，フェロモンの更新について述べる．ACS のフェロモンの更新は，局所更新（式 (12)）と大域更新（式 (13)）の 2 つがある．

$$\tau_{ij} \leftarrow (1-\rho) \times \tau_{ij} + \rho \times \tau_0 \tag{12}$$

$$\tau_{ij} \leftarrow (1-\rho) \times \tau_{ij} + \frac{\rho}{f(s_{\mathrm{gb}})} \quad \forall (i,j) \in s_{\mathrm{gb}} \tag{13}$$

局所更新は蟻が都市を移動するごとに行われ，蟻が移動した辺のフェロモンの値を τ_0 に近づける働きがある．大域更新は一般的な ACO と同様に，すべての蟻が解を作成した後に行われる．フェロモンの大域更新は，s_{gb} に含まれる辺のみに対して行われ，それ以外の辺のフェロモンの値は減らない．よって，τ_0 は MMAS の $\tau_{\min}$ に，$f(s_{\mathrm{gb}})$ の逆数は MMAS の $\tau_{\max}$ に相当し，探索が進むにつれて重要な辺のフェロモンの値が大きくなる．

2.5　関連研究

2.5.1　フェロモンの初期値の偏りによる探索の高速化

通常の TSP を対象に，ACS のフェロモンの初期化を改良した研究 [12], [13]，MMAS のフェロモンの初期化を改良した研究 [14], [15] がある．

Tsai らは，NN 法を改良した Dual Nearest Neighbor（DNN）法を提案し，ACS に対して DNN 法を用いてフェロモンの初期状態に偏りを与えた [12]．DNN 法は，まず都市をランダムに 1 つ選択し，次にその都市から最も離れた都市を選ぶ．この 2 つの都市を出発都市として，すべての都市が訪問されるまで，交互に最も近い未訪問都市に移動する．よって，DNN 法では 2 つの経路が作成される．TDTSP を対象とする場合，最初の都市から最も離れた都市を出発する時間が決められないため，DNN 法は TDTSP に適用できない．また，Tsai らは DNN 法の提案のほかに，NN を複数回実行し（MNN 法），複数の初期解を用いてフェロモンに偏りを与える手法を提案し，DNN 法との比較を行った．この手法は，初期解ごとに含まれている辺のフェロモンの値を τ_0 だけ増やすことで，フェロモンに偏りを与えている．

Dai らは，ACS に対して最小全域木（MST）を用いてフェロモンの初期状態に偏りを与えた [13]．フェロモンの初期化は式 (14) で行う．

$$\tau_{ij} = \begin{cases} [\tau_0]^{1/\theta} & \text{if } (i,j) \in \mathrm{MST} \\ \tau_0 & \text{otherwise} \end{cases} \tag{14}$$

θ は MST の重みを表す 1 以上のパラメータである．τ_0 は 1 未満の値であるため，MST に含まれる辺のフェロモンの値は τ_0 より大きくなる．

著者らは，MNN 法と 2-opt [18] で複数の局所最適解を初期解として作成し，初期解の精度に基づいてフェロモンの初期状態に偏りを与える手法を提案した [14], [15]．この手法は MMAS をベースとしており，フェロモンの初期化は式 (15)，(16) で行う．

$$\tau_{ij} = (1-r) \times \tau_0 + \frac{r}{n_0} \times \sum_{k=1}^{n_0} \delta\tau_{ij}^k \tag{15}$$

$$\delta\tau_{ij}^k = \begin{cases} 1/f(s_0^k) & \text{if } (i,j) \in s_0^k \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

τ_0 は MMAS のフェロモンの初期値（式 (4)），n_0 は初期解の数，s_0^k は k 番目の初期解，r は範囲 $[0,1]$ の初期解の重みである．r が 1 に近いほど，初期解を用いて追加されるフェロモンの値が大きくなる．

Tsai らの手法も MNN 法で初期解集合を作成するが，本手法は初期解の精度を考慮してフェロモンの初期値に偏りを与える点と，初期解の重みを考慮している点で，Tsai らの手法と異なる．

2.5.2　時間依存問題の広域道路網への適用例

TDTSP を広域道路網に適用するためには，都市間の経路から都市間の旅行時間を計算する必要がある．TDTSP を広域道路網に適用した研究は見られないが，時間依存配送計画問題を広域道路網に適用した研究として，遺伝的アルゴリズムに関する研究 [16]，ACO に関する研究 [17] がある．どちらの研究も，広域道路網への適用実験では最短経路問題を解いて旅行時間を計算しているが，この計算は解の探索前に行われており，計算時間に関して言及されていない．現実問題への適用を考えた場合，都市と時間に関する全通りの経路と旅行時間を求めることは，計算時間と記憶容量に関して無駄であるため，提案手法では必要に応じて最短経路問題を解く．

3.　提案手法

3.1　フェロモンの初期状態の偏りによる探索領域の削減

TDTSP は，ACO のように逐次的に解を構築する手法との相性が良く，安定して精度の良い解を発見できる ACO として MMAS がある．よって，提案手法では MMAS を用いて解探索を行う．時間依存配送計画問題を対象とした ACO の研究 [17], [21] や，フェロモンの初期状態に偏りを与える研究 [12], [13] など，ACS に関する研究も数多くある．ACS の特徴として収束の早さがあるが，MMAS と比べると解の精度が低下する場合がある．MMAS は安定して精度の良い解を発見できるが，他の ACO と比べて収束に時間がかかるという欠点がある．TDTSP を対象問題とした場合，旅行時間の変化間隔で精度の良い解を発見する必要があるため，MMAS の解の精度を落とすことなく収束速度を向上させることを目指す．

通常の TSP を対象に，MMAS のフェロモンの初期状態に偏りを与える手法 [14], [15] では，MNN 法と 2-opt により複数の局所最適解を初期解集合として，初期解集合に含

まれる辺のフェロモンの値を増加させている．通常の TSP を対象とした 2-opt では，変更する辺の距離の差だけを見ることで解の評価が可能である．一方，TDTSP を対象とする場合，変更が生じる都市以降の経路に関して，総旅行時間を計算し直す必要があり計算量が大きくなる．よって，提案手法では 2-opt を用いず，MNN 法のみで初期解集合を作成する．

TDTSP は出発都市が固定されているため，通常の TSP を対象とした MNN 法では 1 通りの解しか作成できない．そこで，出発都市から最初に訪問する都市を初期解ごとに変更し，出発都市から 2 番目に訪問する都市以降は通常の NN 法と同様に選択することで，$(n-1)$ 通りの解を作成できるように MNN 法を改良した．提案手法で用いた初期解集合の作成方法（MNN-TDTSP）を**図 3** に示す．

3.2　TDTSP を対象とした MMAS

TDTSP は解の総旅行時間を最小にすることが目的であるため，蟻による解の作成で用いるヒューリスティック値（式 (5)）は，都市間の旅行時間の逆数とする．蟻 k が都市 i を時間 t_i に出発するとき，都市 i，j 間のヒューリスティック値を式 (17) に示す．

$$\eta_{ij} = \frac{1}{T_{ij}(t_i)} \tag{17}$$

提案手法の処理の流れは**図 4** となる．

3.3　広域道路網への適用

本論文で広域道路網を考えるとき，道路網の頂点を交差点，道路網の辺をリンク，リンクの旅行時間をリンク旅行時間と表現する．また，TDTSP の出発都市をデポ，出発都市以外の都市を顧客と表現し，デポまたは顧客は交差点上に設定されているものとする．

本研究では，顧客間の経路と旅行時間を高速に求めるため，リンク旅行時間が変化する Δt ごとにダイクストラ法 [20] を実行し，求めた経路と旅行時間を保持する．ただし，顧客と時間に関するすべての組合せに対してダイクストラ法を実行するのは現実的ではないため，顧客間の旅行時間が必要になった時点で，その顧客間の旅行時間が計算されていない場合にダイクストラ法を実行する．顧客間の旅行時間の計算方法を**図 5** に示す．図 5 における n_{cl} は，通常の TSP を対象とする ACO で用いられている Candidate List [1] に関するパラメータである．Candidate List とは，顧客ごとに近い顧客を上位 n_{cl} 個並べたものであり，利用するためにはソートが必要になるが，提案手法ではダイクストラ法がソートの役割を果たしている．よって，ダイクストラ法を実行したとき，顧客間の経路と旅行時間のほかに，Candidate List も保持する．

procedure MNN - TDTSP ( )
  for ($k = 2$ to n) **do**
    $s_0^{k-1} = (1)$　// 1は出発都市
    $s_0^{k-1} \leftarrow s_0^{k-1} \otimes k$　// 2番目の都市としてkをs_0^{k-1}に追加
    $N' = N - \{1, k\}$　// 未訪問都市集合
    $i = k$
    $t = T_{1i}(0)$　// 2番目の都市の出発時間
    while (N' is not empty) **do**
      $j = \arg\min_{l \in N'} T_{il}(t)$　// 旅行時間に関する最近隣都市
      $s_0^{k-1} \leftarrow s_0^{k-1} \otimes j$　// jをs_0^{k-1}に追加
      $N' \leftarrow N' - \{j\}$
      $t \leftarrow t + T_{ij}(t)$
      $i = j$
    end - while
    $s_0^{k-1} \leftarrow s_0^{k-1} \otimes 1$　// 出発都市に戻る
    $T(s_0^{k-1}) = t + T_{i1}(t)$
  end - for
end - procedure

図 3　TDTSP 向けに改良した MNN 法
Fig. 3　Multiple nearest neighbor heuristic for TDTSP.

procedure ProposedMethod ( )
  Input TDTSP
  Set parameters
  Construct initial solutions using MNN - TDTSP
  $s_{\mathrm{gb}} = \arg\min_{s_0^k, k \in \{1,\cdots,n-1\}} T(s_0^k)$　// 総旅行時間が最も小さい初期解
  Initialize pheromone trails using (15) and (16)
  while (termination condition not met) **do**
    for ($k = 1$ to m) **do**　// mは蟻の数
      Construct a solution s^k using (5) and (17)
    end - for
    $s_{\mathrm{ib}} = \arg\min_{s^k, k \in \{1,\cdots,m\}} T(s^k)$
    if ($T(s_{\mathrm{ib}}) < T(s_{\mathrm{gb}})$) **then**
      $s_{\mathrm{gb}} = s_{\mathrm{ib}}$
      Calculate $\tau_{\max}$ and $\tau_{\min}$ using (8) and (9)
    end - if
    Update pheromone trails using (6) and (7)
  end - while
end - procedure

図 4　提案手法の処理の流れ
Fig. 4　Proposed method procedure.

4.　評価実験

本章では，TSP のベンチマーク問題を用いた実験（4.2 節），現実の道路網と交通量データを用いた実験（4.3 節）について述べ，提案手法の有効性を確認する．プログラムは Visual C++ 2010 64 bit で作成し，Windows7 64 bit，Core i7-860，16 GB RAM の環境で実験を行った．問題ごと手法ごとの各実験は，乱数のシードを変えて 100 回繰り返し，その結果を用いて評価を行う．

```
procedure CalcTravelTime ( s, g, t, n_cl )
  // sは出発地（デポまたは顧客）
  // gは目的地（デポまたは顧客）
  // tはsの出発時間
  // n_clはCandidate Listのサイズ
  foreach ( node n in real road network ) do
    T[n] = ∞   // tにおけるsからnまでの旅行時間
    d[n] = false   // T[n]が確定した場合にtrue
  end - foreach
  T[s] = 0
  c = 0   // 確定したデポまたは顧客の数
  while ( c ≤ n_cl ) do
    i = arg min_{n,d[n]=false} T[n]
    d[i] = true
    if ( ( i is not s ) and ( i is depot or customer ) ) then
      c++
      T_si(t) = T[i]
      Store the shortest path from s to i at t
      Add i to the candidate list of s at t
    end - if
    foreach ( node j which is adjacent to i ) do
      // l(i, j, t)はtにおけるi, j間のリンク旅行時間
      if ( T[i] + l(i, j, t) < T[j] ) then
        T[j] = T[i] + l(i, j, t)
      end - if
    end - foreach
  end - while
  // gがsのCandidate Listに含まれていない場合は続行
  while ( d[g] is false ) do
    i = arg min_{n,d[n]=false} T[n]
    d[i] = true
    if ( i is depot or customer ) then
      T_si(t) ← T[i]
      Store the shortest path from s to i at t
      if ( i is g ) then
        return
      end - if
    end - if
    foreach ( node j which is adjacent to i ) do
      if ( T[i] + l(i, j, t) < T[j] ) then
        T[j] = T[i] + l(i, j, t)
      end - if
    end - foreach
  end - while
end - procedure
```

図 5　顧客間の旅行時間の計算方法
Fig. 5　Calculation procedure of travel time between customers.

4.1　比較手法

比較手法として，ACS，MMAS，Dai らの手法 [13] を変更したもの（MMAS+mst）を用いた．ACS は，MMAS と比べて解の精度が低下する場合があるが，収束が早い代表的な ACO であることから比較手法に用いた．MMAS は，提案手法のベースとなっている ACO であり，安定して精度の良い解を発見できるが，収束が遅いという問題点がある．

提案手法は，MMAS のフェロモンの初期化方法を改良したものである．よって，フェロモンの初期化法の比較のため，MST をフェロモンの初期化に利用する手法を考える．Dai らの手法は ACS をベースとしていることから，MMAS をベースとして MST を利用できるように，式 (14) の τ_0 を式 (18) で計算した．

$$\tau_0 = \frac{1 - \sqrt[n]{0.05}}{(n/2 - 1) \times \sqrt[n]{0.05}} \times \frac{1}{\rho \times T(s_{\mathrm{NN}})} \tag{18}$$

TDTSP は旅行時間が時間によって変化するため，高速に MST を求めることは困難である．よって，対称グラフの MST を求めるアルゴリズムであるプリム法 [20] と同様に全域有向木を求め，その全域有向木を近似的な MST（MST′）としてフェロモンの初期化に用いた．

各手法のパラメータの値を**表 1** に示す．これらの値は予備実験により求めた．ACO における探索の終了条件は，蟻による解の作成回数を $5000n$ とした．

4.2　TSP のベンチマークによる実験

4.2.1　実験方法

TDTSP の例題として，TSPLIB [19] で公開されている問題の中から**表 2** の 9 問を用いた．これらの問題は小さいサイズの問題であるが，配送車が 1 日に訪問できる顧客の数は 200 ほどであることから，評価実験に適していると考えた．

TSP のベンチマーク問題から式 (1) を用いて TDTSP を作成するため，旅行時間の変化間隔 Δt と旅行時間の変化の大きさ C_f を決める必要がある．日本では道路交通情報通信システム（VICS）により 5 分間隔で最新の交通情報を得ることができる．各問題の最適解から，eil51，eil76，eil101 の旅行時間の単位を分と仮定して Δt を 5 とし，それ以外の問題は単位を秒と仮定して Δt を 300 と設定した．C_f に関しては，東京都中央区 11 km × 9 km の道路網に対し，平日の交通量データを確認した．具体的には，異なる交差点を 300 個ランダムに選んで顧客とし，89700 通りの顧客間に関して，リンク旅行時間が変化するごとにダイクストラ法で顧客間の旅行時間を計算した．その結果から，顧客間の旅行時間の変化率を $|T_{ij}(t_i)/T_{ij}(t_i - \Delta t) - 1|$ として平均変化率を計算した（**図 6**）．この結果から，C_f は 0.1 として TDTSP を作成した．

表 2 における TSP の最適解 s_{opt} の総旅行時間 $T(s_{\mathrm{opt}})$ は，これらの条件で TDTSP を作成し，TSP の最適解を TDTSP で評価した総旅行時間である．NA と表記しているものは，最適解の巡回路が公開されていない問題である．また，表 2 における s_{pb} は，実験で得られた問題ごとの最良解を表している．具体的には，各問題に対して手法ご

表 1　手法ごとのパラメータの値
Table 1　Parameter values of each methods.

パラメータ	ACS	MMAS	MMAS+mst	提案手法
蟻の数	10	n	n	n
α	1	1	1	1
β	5	5	5	5
ρ	0.1	0.02	0.02	0.02
n_{cl}	20	20	20	20
q_0	0.9	NA	NA	NA
θ	NA	NA	1.8	NA
r	NA	NA	NA	0.9

表 2　実験に用いた TSP のベンチマーク問題
Table 2　TSPLIB instances used for experiments.

問題	都市数	Δt	TSP の最適解 s_{opt}		$T(s_{\mathrm{pb}})$
			巡回路長	$T(s_{\mathrm{opt}})$	
eil51	51	5	426	581.93	501.15
eil76	76	5	538	730.64	642.68
eil101	101	5	629	868.61	791.51
kroA100	100	300	21282	27573.54	24908.13
u159	159	300	42080	NA	53962.43
d198	198	300	15780	NA	17905.19
kroA200	200	300	29368	NA	35781.35
pr299	299	300	48191	NA	63491.66
lin318	318	300	42029	NA	54131.37

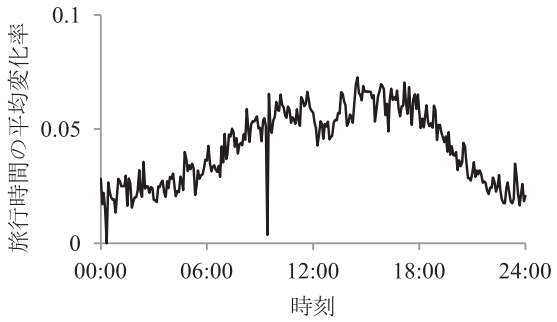

図 6　現実の旅行時間の平均変化率
Fig. 6　Mean change rate of real world travel times.

とに 100 回実験を行っているため，各問題に 400 個の s_{gb} が求まる．400 個の s_{gb} に対して 2-opt を適用し，得られた最良の局所最適解を s_{pb} とした．以降の実験結果では，$(T(s_{\mathrm{gb}})/T(s_{\mathrm{pb}})-1)$ で計算される値を s_{gb} の誤差率として用いる．

4.2.2　MST′ と MNN-TDTSP の比較

MST′ と MNN-TDTSP（提案手法）を比較するために，以下の指標（式 (19)，(20)）を確認した．

$$R_{\mathrm{cover}} = \frac{|A_0 \cap A(s_{\mathrm{pb}})|}{n} \tag{19}$$

$$R_{\mathrm{reduction}} = 1 - \frac{|A_0|}{|A|} \tag{20}$$

$$A_0 = \begin{cases} A(\mathrm{MST}') & \text{for MST}' \\ \bigcup_k A(s_0^k) & \text{otherwise} \end{cases} \tag{21}$$

$A(\mathrm{MST}')$ は MST′ の辺集合，$A(s)$ は解 s の辺集合である．よって，R_{cover} は s_{pb} の辺の見逃しが少ないほど高い値となり，$R_{\mathrm{reduction}}$ はフェロモンの初期値に偏りを与える辺が少ないほど高い値となる．R_{cover} と $R_{\mathrm{reduction}}$ の結果（**表 3**）より，次のことが確認できる．

- MST′ に関して，$|A_0|$ は $(n-1)$ であるため，MST′ の $R_{\mathrm{reduction}}$ は $(n-1)/n$ であり 1 に近い値となる．
- MNN-TDTSP の R_{cover} の値は MST′ の値の約 3 倍であり，MNN-TDTSP は MST′ よりも s_{pb} の辺を発見できている．
- MNN-TDTSP の $R_{\mathrm{reduction}}$ の値は MST′ の値よりも小さく，eil51 では全体の辺の約 2 割が MNN-TDTSP に含まれている．
- MNN-TDTSP の $R_{\mathrm{reduction}}$ の値から，MNN-TDTSP の $|A_0|$ は MST′ の $|A_0|$ と比較して，約 16 倍から 50 倍大きい．

R_{cover} と $R_{\mathrm{reduction}}$ はトレードオフの関係にあり，R_{cover} は解の精度に，$R_{\mathrm{reduction}}$ は収束の早さに影響する．たとえ $R_{\mathrm{reduction}}$ が大きかったとしても，R_{cover} が小さい場合には得られる解の精度が低下してしまう．したがって，MMAS+mst と提案手法を比較すると，MMAS+mst の方が収束は早いが，提案手法は MMAS+mst よりも精度の良い解を発見できると予想される．

4.2.3　探索終了時の最良解における比較

まず，各手法の探索が収束していることを確認するため，問題サイズが最大の lin318 に対して，s_{gb} の誤差率の推移と λ-branching factor [1] の推移を**図 7** に示す．λ-branching factor は ACO のフェロモンの状況を評価する指標であり，TDTSP に対しては範囲 $[1, n-1]$ の実数をとる．λ-branching factor が 1 に近いほど一部の辺のフェロモンの値が相対的に大きくなっており，ACO の探索領域が削減されていることを表す．図 7 より，探索の終了条件において，各手法は探索が収束していると判断する．

次に，探索終了時の s_{gb} の平均誤差率の百分率を**表 4** に示す．表 4 の括弧内の数値は標準偏差である．表 4 より，次のことが確認できる．

- ACS と MMAS の s_{gb} の誤差率を比較すると，eil76 では約 2 倍，u159 では約 3 倍 ACS の値が大きく，その他の問題では同等である．
- ACS と MMAS の標準偏差を比較すると，すべての問題で ACS の値が大きく，ACS は MMAS よりも安定していない．
- MMAS+mst と MMAS を比較すると，MMAS+mst は 6 問で s_{gb} の誤差率が MMAS より大きい．
- 提案手法と MMAS を比較すると，s_{gb} の誤差率，標準

表 3　MST′ と MNN-TDTSP の比較
Table 3　Comparison between MST′ and MNN-TDTSP.

問題	MST′		MNN-TDTSP	
	R_{cover}	$R_{\mathrm{reduction}}$	R_{cover}	$R_{\mathrm{reduction}}$
eil51	0.25	0.98	0.93	0.83
eil76	0.33	0.99	0.94	0.87
eil101	0.41	0.99	0.94	0.88
kroA100	0.37	0.99	0.94	0.92
u159	0.42	0.99	0.96	0.92
d198	0.32	0.99	0.97	0.95
kroA200	0.44	1.00	0.95	0.94
pr299	0.35	1.00	0.97	0.94
lin318	0.36	1.00	0.96	0.95

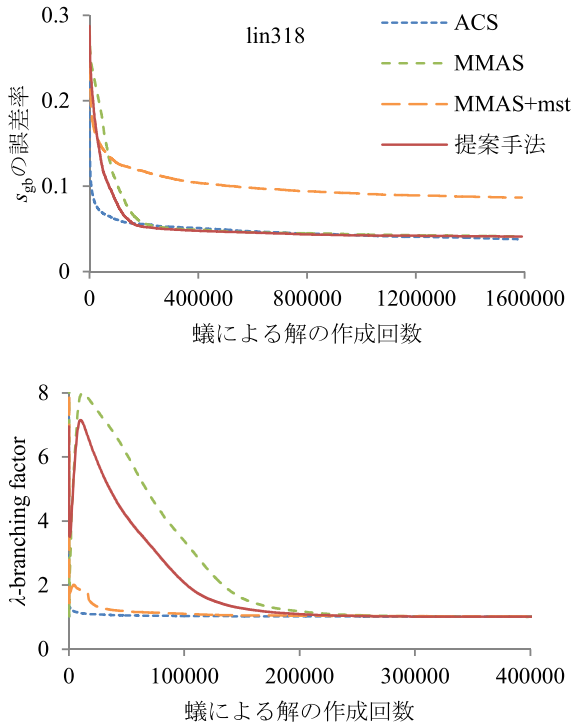

図 7　探索の収束の確認
Fig. 7　Verification of convergence in search.

表 4　探索終了時の s_{gb} の平均誤差率の百分率（括弧内は標準偏差）
Table 4　Percent mean error rate of s_{gb} upon termination of search. The values in parenthesis are the standard deviation.

問題	ACS	MMAS	MMAS+mst	提案手法
eil51	3.85 (1.73)	3.12 (1.28)	4.23 (2.02)	3.17 (1.29)
eil76	3.00 (2.40)	1.72 (0.55)	5.42 (2.30)	1.82 (0.61)
eil101	4.51 (1.83)	5.52 (1.08)	5.63 (2.13)	5.50 (1.15)
kroA100	2.95 (2.27)	1.89 (0.50)	5.70 (2.01)	1.99 (0.60)
u159	5.81 (3.23)	2.05 (0.68)	7.28 (2.60)	1.99 (0.73)
d198	4.87 (1.68)	5.25 (0.16)	6.70 (2.39)	5.33 (0.26)
kroA200	2.99 (1.78)	2.23 (0.75)	7.24 (2.26)	2.25 (0.62)
pr299	4.77 (2.56)	4.96 (1.21)	8.80 (2.12)	4.92 (1.16)
lin318	3.79 (1.98)	4.14 (1.42)	8.65 (2.84)	4.10 (1.23)

偏差ともに同等である．

MMAS+mst の s_{gb} の誤差率は，6 個の問題（eil76, kroA100, u159, kroA200, pr299, lin318）で MMAS のものよりも明らかに大きい．MMAS+mst と MMAS の違いはフェロモンの初期化のみであるため，MST′ の R_{cover}（表 3）が小さいためであると考えられる．一方，提案手法は MMAS と同等の s_{gb} の誤差率を得ていることから，フェロモンの初期化が悪影響を与えていないことが確認できる．

4.2.4　近似解による収束速度向上の確認

まず，手法ごとの探索の様子を確認するため，CPU 時間［秒］ごとの s_{gb} の誤差率の平均値を**図 8** に示す．図 8 では，MMAS が収束したと判断できる CPU 時間までの結果を示しており，次のことが確認できる．

- 誤差率 0.2 に到達する CPU 時間は，ACS, MMAS+mst, 提案手法，MMAS の順で小さい．
- 提案手法は MMAS と比べ，すべての問題に対して収束が早まっている．
- MMAS+mst は誤差率 0.1 に到達していない問題がある．

MMAS+mst は $R_{\mathrm{reduction}}$ が大きいため収束が早いが，R_{cover} が小さいため MMAS より解の精度が低下している．一方，提案手法は MMAS+mst より収束は遅いが R_{cover} が大きいため，MMAS と比べて解の精度を落とさずに収束を早められている．収束速度の比較に関しては，図 7 の λ-branching factor の結果からも見て取れる．

次に，s_{gb} の誤差率が 0.10，0.05 となる CPU 時間［秒］の平均値（標準偏差）を**表 5** に示す．表 5 でのハイフンは探索失敗を意味しており，探索終了時の s_{gb} の誤差率が 0.10 または 0.05 より大きい場合が 1 回でもあった場合に該当する．現実の交通量データには計測誤差が 10%以上含まれており，本来は予測交通量にも誤差が含まれる．巡回路の総旅行時間では誤差が平均化されることも考え，s_{gb} の誤差率が 0.10，0.05 となる CPU 時間を確認した．表 5 より次のことが確認できる．

- ACS で誤差率 0.10 の近似解が求まる問題（eil51, eil101, lin318）に対しては，必要な CPU 時間は ACS が最も小さい．
- MMAS+mst は，eil51 の近似解（誤差率 0.10）のみ安定して求めることができた．
- MMAS と提案手法は，すべての問題に対して誤差率 0.10 の近似解が求まっており，誤差率 0.05 の近似解が求まっている問題もある．
- MMAS で求められている近似解は，提案手法でも求められている．
- 提案手法は MMAS と比較すると，探索に必要な CPU 時間が約 0.55 倍になっている．

以上から，提案手法は MMAS と同様に安定して精度の

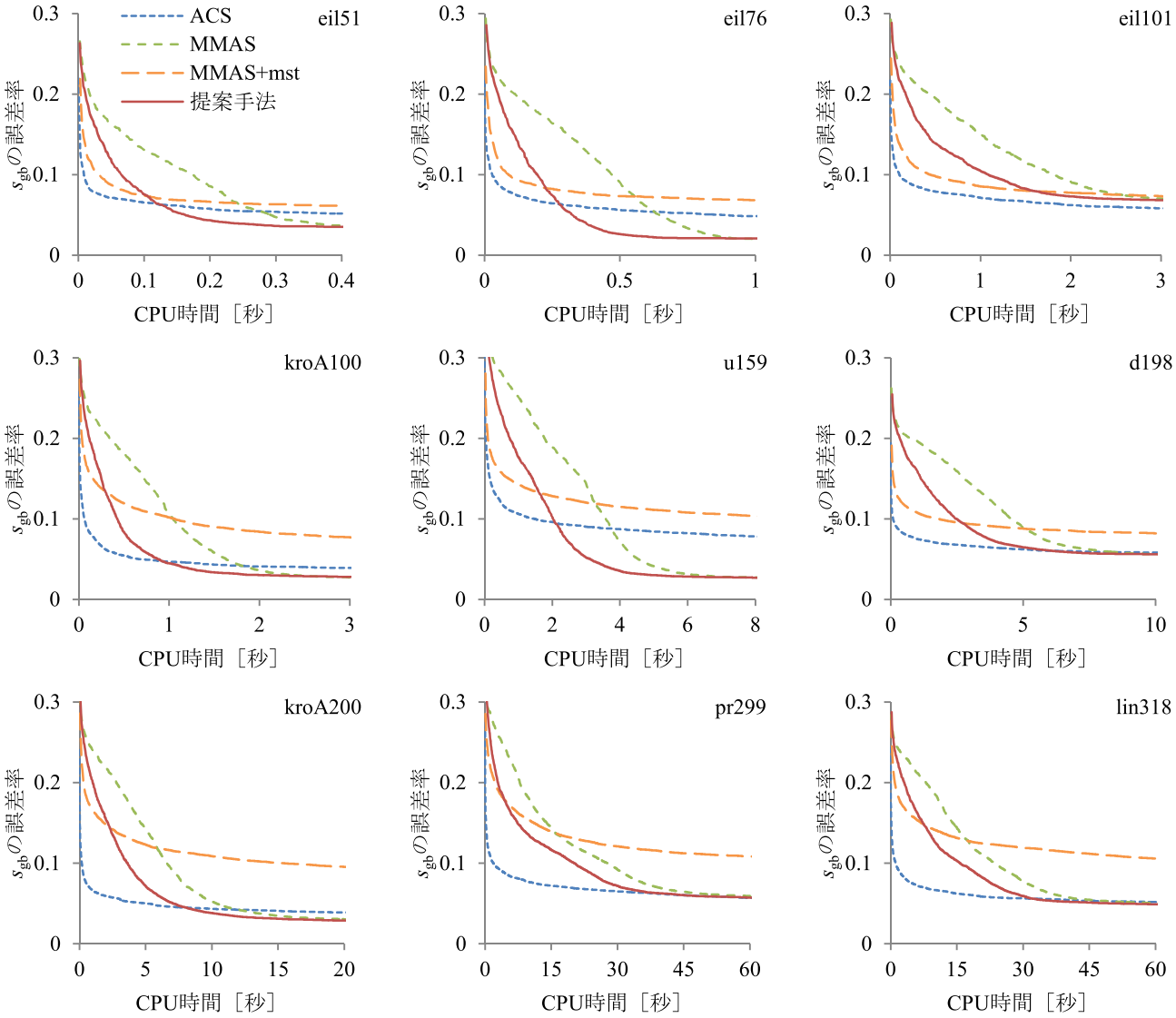

図 8　CPU 時間ごとの s_{gb} の誤差率の平均値

Fig. 8　Mean error rate of s_{gb} in each CPU time.

表 5　近似解を求めるために必要な CPU 時間

Table 5　CPU time required to find approximate solutions. The values in parenthesis are the standard daviation.

問題	ACS		MMAS		MMAS+mst		提案手法	
	0.10	0.05	0.10	0.05	0.10	0.05	0.10	0.05
eil51	0.03 (0.11)	-	0.17 (0.06)	-	0.11 (0.25)	-	0.07 (0.03)	-
eil76	-	-	0.48 (0.09)	0.64 (0.10)	-	-	0.20 (0.06)	0.32 (0.10)
eil101	0.36 (0.69)	-	1.82 (0.28)	-	-	-	1.09 (0.26)	-
kroA100	-	-	1.05 (0.15)	1.61 (0.26)	-	-	0.42 (0.09)	0.99 (0.69)
u159	-	-	3.62 (0.38)	-	-	-	2.11 (0.42)	3.27 (2.34)
d198	-	-	4.58 (0.36)	-	-	-	2.49 (0.42)	-
kroA200	-	-	6.58 (0.53)	10.42 (1.51)	-	-	3.68 (0.48)	7.18 (1.22)
pr299	-	-	28.27 (3.25)	-	-	-	20.81 (3.02)	-
lin318	7.19 (31.07)	-	24.06 (2.71)	-	-	-	16.35 (2.78)	-

良い解を求めつつ，収束を早めていることが確認できる．

4.3　現実の道路網と交通量データによる実験

4.3.1　実験方法

実験対象の道路網は，日本で最も交通量が多い東京都中

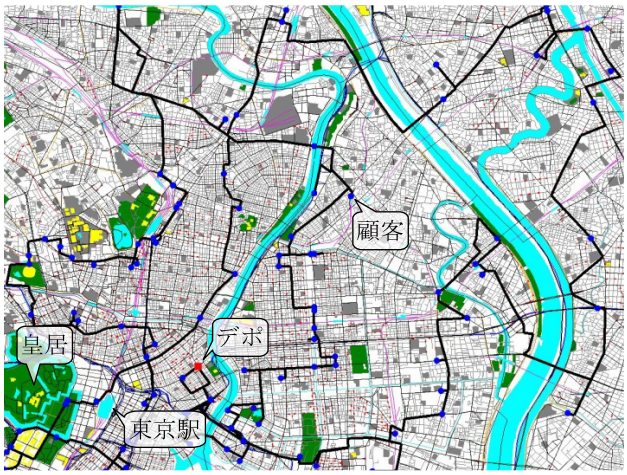

図 9　実験対象の道路網
Fig. 9　Real road network used for experiments.

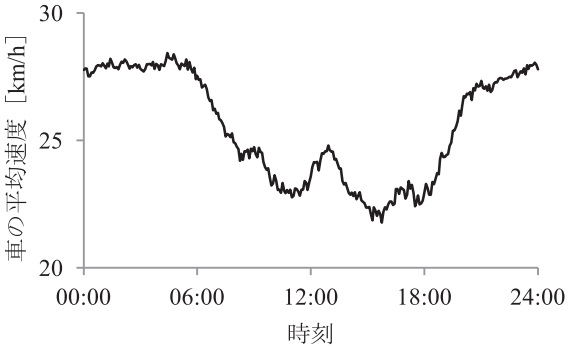

図 10　車の平均速度の変化
Fig. 10　Variation of average vehicle speed.

央区 11 km × 9 km（**図 9**）とし，交通量データは 2003 年 6 月 17 日（**図 10**）から 19 日のものを用いた．具体的なデータは，道路網はカーナビに用いられているナビ研 S 規格地図，交通量データは VICS で提供されている 5 分間隔の平均リンク旅行時間である．予測交通量は解探索前に既知であると仮定しているので，交通量データを予測交通量として用いた．現実問題では予測交通量に誤差が含まれるが，本論文では計算時間に関して評価を行っているため，実験結果において一般性を失わない．

デポと顧客は重複がないようにランダムに交差点上に設定し，顧客数 50，100，200，300 の 4 つの問題を作成した．また図 10 による交通量の変化から，デポを出発する時刻を 6:00，12:00，18:00 とし，顧客数と出発時刻による 12 個の問題に対して実験を行った．図 9 は顧客数 100 の例を示しており，赤い正方形はデポを，青い丸は顧客を，黒い太線は出発時刻 12:00 における s_{pb} を表している．実験条件は 4.2 節と同様のものを用いた．

4.3.2　実験結果

ベンチマーク問題に対する実験と同様に，出発時刻 6:00 における CPU 時間［秒］ごとの s_{gb} の誤差率の平均値を**図 11** に，s_{gb} の誤差率が 0.10，0.05 となる CPU 時間［秒］の平均値（標準偏差）を**表 6** に示す．図 11 より，次のことが確認できる．

- 収束に関しては，ACS, MMAS+mst, 提案手法, MMAS

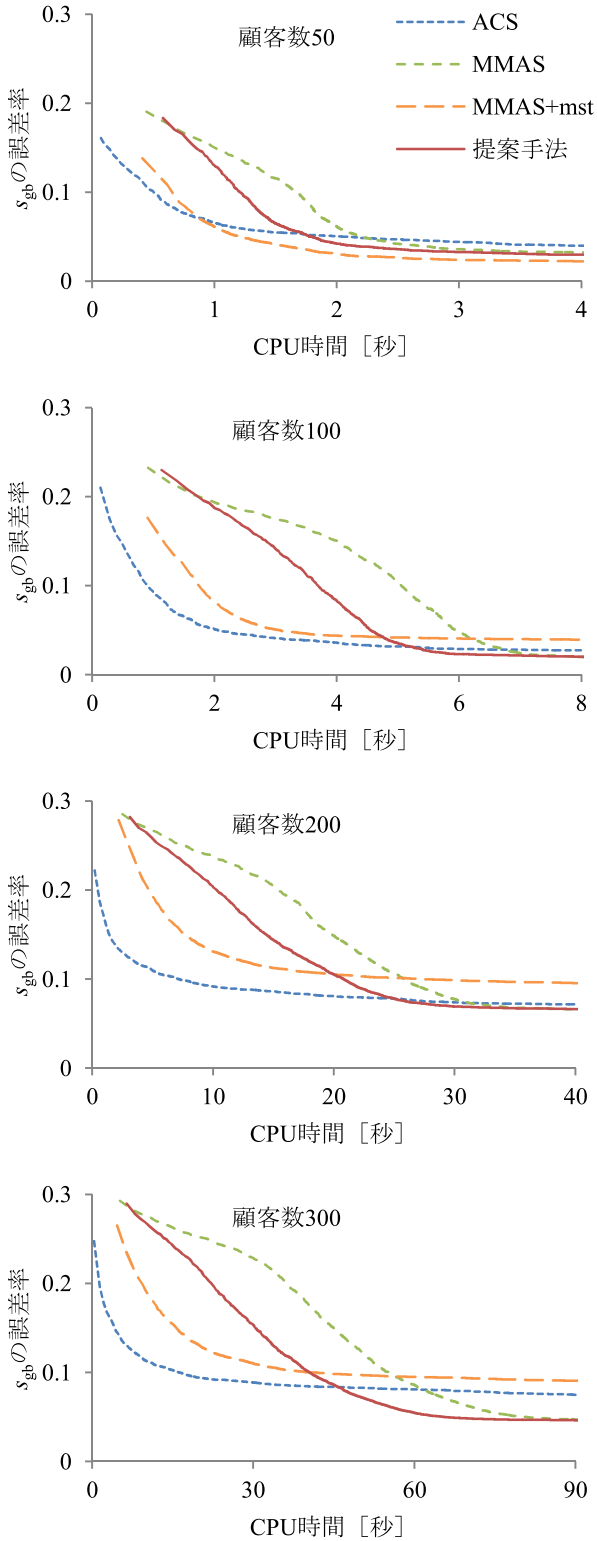

図 11　広域道路網における CPU 時間ごとの s_{gb} の誤差率の平均値（出発時刻 6:00）
Fig. 11　Mean error rate of s_{gb} in each CPU time in a case of real road network. The departure time form the depot is 6:00 a.m.

表 6　近似解を求めるために必要な CPU 時間（広域道路網）
Table 6　CPU time required to find approximate solutions in a case of real road network. The values in parenthesis are the standard daviation.

顧客数	出発時間	$T(s_{pb})$	ACS		MMAS		MMAS+mst		提案手法	
			0.10	0.05	0.10	0.05	0.10	0.05	0.10	0.05
50	6:00	11421	0.61 (0.24)	-	1.67 (0.12)	2.26 (0.55)	0.66 (0.12)	-	1.23 (0.11)	1.62 (0.11)
	12:00	11783	0.67 (0.39)	-	1.69 (0.18)	-	0.80 (0.16)	-	1.32 (0.11)	-
	18:00	10772	0.48 (0.36)	-	1.42 (0.13)	1.92 (0.20)	0.57 (0.04)	-	1.09 (0.09)	1.33 (0.07)
100	6:00	16215	1.07 (0.57)	-	5.09 (0.28)	5.96 (0.32)	1.89 (0.37)	-	3.67 (0.26)	4.42 (0.20)
	12:00	16773	-	-	5.18 (0.20)	7.70 (3.68)	-	-	3.83 (0.29)	4.97 (0.27)
	18:00	14816	0.98 (0.53)	-	4.46 (0.24)	6.10 (0.77)	1.69 (0.15)	-	3.23 (0.21)	4.27 (0.23)
200	6:00	23902	17.48 (26.66)	-	25.48 (1.72)	-	-	-	20.65 (1.64)	-
	12:00	25197	2.66 (2.06)	-	19.54 (1.13)	27.78 (1.52)	9.17 (6.28)	-	14.08 (0.95)	20.79 (1.29)
	18:00	21771	1.75 (0.55)	-	19.37 (1.03)	32.39 (13.53)	7.36 (1.04)	-	13.33 (0.99)	19.07 (1.13)
300	6:00	31752	-	-	55.65 (2.95)	76.26 (6.77)	-	-	40.77 (2.86)	57.90 (3.36)
	12:00	31406	7.59 (6.48)	-	47.97 (2.44)	73.28 (9.43)	-	-	34.41 (1.91)	55.03 (2.77)
	18:00	27827	5.30 (2.61)	-	45.68 (2.37)	73.98 (8.07)	-	-	32.34 (2.25)	49.72 (2.54)

の順で早い．

- ダイクストラ法を実行しているため，ベンチマーク問題よりも計算時間が必要になっている．

表 6 より，次のことが確認できる．

ACS で近似解が求まる場合は，必要な CPU 時間は ACS が最も小さい．

- 安定して求められた近似解の数は，MMAS と提案手法が最も多く，MMAS+mst が最も少ない．
- MMAS で求められている近似解は，提案手法でも求められている．
- 提案手法は MMAS と比較すると，探索に必要な CPU 時間が約 0.7 倍になっている．

以上から，提案手法は現実の道路網と交通量データに対しても有効であることが確認できる．

5. おわりに

本論文では，予測交通量に基づく ACO による TDTSP の解法を提案した．提案手法は，安定して精度の良い解を求めることができる MMAS をベースとし，フェロモンの初期値に偏りを与えることで探索領域を狭め，収束を早めるものである．複数の初期解に基づいてフェロモンの初期化を行うため，TDTSP を対象とした場合でも MNN 法で多様な初期解を作成できるように MNN 法を改良した．提案手法の有効性を確認するため，TSP のベンチマーク問題から TDTSP を作成した実験，現実の道路網と交通量データを用いた実験を行い，提案手法は解の精度を落とさずに収束を早めていることを確認した．

今後の課題として，現実の道路網と交通量データを系統化し，道路網や交通量データの特徴の違いによる探索効率の変化を確認することが考えられる．

謝辞　本研究は JSPS 科研費 23500169 の助成を受けたものです．

参考文献

[1] Dorigo, M. and Stützle, T.: *Ant Colony Optimization*, MIT Press (2004).
[2] Dorigo, M., Birattari, M. and Stützle, T.: Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique, *IEEE Computational Intelligence Magazine*, Vol.1, No.4, pp.28–39 (2006).
[3] Mullen, J.R., Monekosso, D., Barman, S. and Remagnino, P.: A review of ant algorithms, *Expert Systems with Applications*, Vol.36, No.6, pp.9608–9617 (2009).
[4] Dorigo, M. and Stützle, T.: Ant Colony Optimization: Overview and Recent Advances, Handbook of Metaheuristics, *International Series in Operations Research & Management Science*, Vol.146, pp.227–263, Springer US (2010).
[5] Gouveia, L. and Vob, S.: A classification of formulations for the (time-dependent) traveling salesman problem, *European Journal of Operational Research*, Vol.83, No.1, pp.69–82 (1995).
[6] Guntsch, M. and Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP, *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, Vol.2037, pp.213–222, Springer Berlin Heidelberg (2001).
[7] Eyckelhof, J.C. and Snoek, M.: Ant System for a Dynamic TSP: Ant Caught in a Traffic Jam, *Ant Algorithms*, Lecture Notes in Computer Science, Vol.2463, pp.88–99, Springer Berlin Heidelberg (2002).
[8] Mavrovouniotis, M. and Yang, S.: An Immigrants Scheme Based on Environmental Information for Ant Colony Optimization for the Dynamic Traveling Salesman Problem, *Artificial Evolution*, Lecture Notes in Computer Science, Vol.7401, pp.1–12, Springer Berlin Heidelberg (2012).
[9] Kanoh, H. and Ochiai, J.: Solving Time-Dependent Traveling Salesman Problems using Ant Colony Opti-

mization Based on Predicted Traffic, *Proc. International Symposium on Distributed Computing and Artificial Intelligence, Advances in Intelligent and Soft Computing*, Vol.151, pp.25–32, Springer Berlin Heidelberg (2012).
[10] Stützle, T. and Hoos, H.H.: MAX-MIN Ant System, *Future Generation Computer System*, Vol.16, No.8, pp.889–914 (2000).
[11] Dorigo, M. and Gambardella, M.L.: Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Trans. Evolutionary Computation*, Vol.1, No.1, pp.53–66 (1997).
[12] Tsai, C.-F., Tsai, C.-W. and Tseng, C.-C.: A new hybrid heuristic approach for solving large traveling salesman problem, *Information Sciences*, Vol.166, pp.67–81 (2004).
[13] Dai, Q., Ji, J. and Liu, C.: An Effective Initialization Strategy of Pheromone for Ant Colony Optimization, *Proc. 4th International Conference on Bio-Inspired Computing*, pp.1–4, IEEE (2009).
[14] Kanoh, H. and Kameda, Y.: Pheromone Trail Initialization with Local Optimal Solutions in Ant Colony Optimization, *Proc. 2nd International Conference on Soft Computing and Pattern Recognition*, pp.338–343, IEEE (2010).
[15] Kanoh, H., Ochiai, J. and Kameda, Y.: Pheromone Trail Initialization with Local Optimal Solutions in Ant Colony Optimization, *International Journal of Knowledge-Based and Intelligent Engineering Systems*, accepted (2013).
[16] Haghani, A. and Jung, S.: A dynamic vehicle routing problem with time-dependent travel times, *Computers & Operations Research*, Vol.32, No.11, pp.2959–2986 (2005).
[17] Donati, V.A., Montemanni, R., Casagrande, N., Rizzoli, E.A. and Gambardella, M.L.: Time dependent vehicle routing problem with a multi ant colony system, *European Journal of Operational Research*, Vol.185, No.3, pp.1174–1191 (2008).
[18] Gutin, G. and Punnen, P.A.: The traveling salesman problem and its variations, *Combinatorial Optimization*, Vol.12, Springer US (2007).
[19] Reinelt, G., et al.: TSPLIB, available from ⟨http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/⟩ (accessed 2013-11-10).
[20] Korte, B. and Vygen, J.: Combinatorial Optimization: Theory and Algorithms, *Algorithms and Combinatorics*, Vol.21, Springer Berlin Heidelberg (2000).
[21] Montemanni, R., Gambardella, M.L., Rizzoli, E.A. and Donati, V.A.: Ant Colony System for a Dynamic Vehicle Routing Problem, *Journal of Combinatorial Optimization*, Vol.10, No.4, pp.327–343 (2005).

落合 純一　（学生会員）

2009 年筑波大学第三学群情報学類卒業．2011 年同大学大学院システム情報工学研究科博士前期課程修了（修士）．現在，同大学院システム情報工学研究科博士後期課程在学中．アントコロニー最適化法に関する研究に従事．2011 年情報処理学会数理モデル化と問題解決研究会プレゼンテーション賞受賞．

狩野 均　（正会員）

1978 年筑波大学第一学群自然学類卒業．1980 年同大学大学院理工学研究科修了．同年日立電線（株）入社．同社オプトロシステム研究所において人工知能・ニューラルネットワークの応用に関する研究に従事．1993 年より筑波大学．現在，同大学システム情報系教授．工学博士．知識処理，進化計算，人工生命，高度交通システムの研究に従事．1992 年電気学会論文賞，1999 年 WSC4 国際会議論文賞，2003 年・2006 年情報処理学会優秀研究報告賞各受賞．人工知能学会，IEEE 各会員．