情報処理学会論文誌 Vol.53 No.11 2399–2408 (Nov. 2012)

# 不均一並行計算環境を想定した非同期アントコロニー最適化法

八槇 博史[1,a)] 中村 悟[2] 佐藤 亮介[2]

受付日 2012年1月27日, 採録日 2012年7月2日

**概要**：アントコロニー最適化法において複数のアリを並列に動作させる際，従来方式では共有オブジェクトであるフェロモン行列に関する共有処理が同期的に行われるため，アリの探索時間にバラつきがある場合に他のアリの探索終了を待つなどのボトルネックがある．本研究においては，探索の終了したアリからの情報を随時フェロモン行列に反映させることによって更新における待ち時間を削減する非同期アントコロニーシステムを考案した．フェロモン行列の値に関するシステム内での一貫性が保証できない場合でも収束が保証されることを証明し，また並列計算機を用いた実験により速度向上が見られることを確認した．

**キーワード**：アントコロニー最適化法，並列計算，非同期アルゴリズム，AACS

# An Asynchronous Ant Colony Optimization Algorithm for Heterogeneous Concurrent Computing Environments

Hirofumi Yamaki[1,a)] Satoshi Nakamura[2] Ryosuke Sato[2]

Received: January 27, 2012, Accepted: July 2, 2012

***Abstract:*** Traditional Ant Colony Optimization algorithms suffer from bottle-necks especially when multiple ants with different length of calculation speed run in parallel. This is because the pheromone table in a colony, which is an object shared by all the ants in it, has to be updated synchronously and causes ants to wait. In this research, we propose an asynchronous ACO algorithm that allows pheromone tables to be updated at any time without waiting for all the ants to finish searching paths. This paper gives a convergence proof and the results of experiments.

***Keywords:*** ant colony optimization, parallel computing, asynchronous algorithm, AACS

## 1. 序論

近年，計算機の規模・複雑性・能力の急速な増大にともなって，最適化問題の解法であるメタヒューリスティクスへの注目が高まってきている [22]．その多くは，物理現象，生命・生物現象，社会現象などにおける類推を基礎としており，代表的なものとしては，遺伝的アルゴリズム（Genetic Algorithm, GA）[8]，シミュレーテッド・アニーリング（Simulated Annealing, SA）[9]，などがよく知られている．また，初期のマルチエージェントシステム研究の成果である市場指向プログラミング（Market-Oriented Programming, MOP）[18] や，本論文で主に扱うアントコロニー最適化法（Ant Colony Optimization, ACO）[4] もメタヒューリスティクスの一種ととらえられる．

解析的に解くことが困難な非線形の事象を計算機シミュレーションによって解析するアプローチと，メタヒューリスティクスとを組み合わせることによって，従来の解析的手法のみでは困難であった，パラメータの次元数（自由度）が大きく，かつ複雑なプロセスを含むシステムの設計が可能となってきている．そこでは，1 つのパラメータセットに対して 1 回のシミュレーション実行が行われ，その結果

[1] 名古屋大学情報基盤センター
Information Technology Center, Nagoya University, Nagoya, Aichi 464–8601, Japan
[2] 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University, Nagoya, Aichi 464–8601, Japan
[a)] yamaki@itc.nagoya-u.ac.jp

得られる性能などが最良となるパラメータが，膨大な数の試行を繰り返すことによって探索される．応用としては，構造設計 [21], [24] や知能ロボットの行動学習 [16] などでの事例が報告されている．

このような背景により，近年発達してきたスーパコンピュータ，グリッド・コンピューティング，クラウド・コンピューティングなどの大規模計算環境上でメタヒューリスティクスを利用するニーズが高まっている．特に後二者の特徴として，プロセッサ間の遅延が従来の大規模並列計算機と比較して非常に大きい（疎結合）ことと，プロセッサ間に性能のばらつきのある（不均一）構成であることがあげられる．また，スーパコンピュータにあっても，東京工業大学による TSUBAME では不均一構成が採られるなど，従来のように密結合の均一構成を前提とはしがたくなってきている [6]．

その一方で，メタヒューリスティクスの並列化においては，探索途中の情報をシステム内で共有するための同期通信が必要となり，1 回の評価に要する時間にばらつきがある場合には，システム内で大量の待ち時間が発生して計算効率の低下が起きる．上述のような疎結合で不均一な計算機環境では特に重大となる．

メタヒューリスティクスの並列化に関しては，特に GA においてはさかんに研究されている．必ずしも上記のような不均一な計算機環境での問題を想定したものばかりではないにせよ，並列遺伝的アルゴリズム（Parallel Genetic Algorithm, PGA）[12] など，母集団全体に対してではなく分割されたサブ集団に対して選択を行うアルゴリズムについて，多くの蓄積がすでになされている．Belding [1] が実験により示した，プロセッサ間通信の削減による計算時間の短縮は，不均一な計算機環境についても適用可能な成果であると考えられる．

一方で，メタヒューリスティクス全体について考えると，このような並列化に関する検討が十分に進んでいるとはいいがたい．マルチエージェントシミュレーション研究の文脈で，様々なメタヒューリスティクスを組み合わせて利用しているという観点からは，GA におけるのと同様の成果が各々のアプローチにおいても得られることが望ましい．本論文では特に ACO に着目し，その一種であるアントコロニーシステム（Ant Colony System, ACS）[4] をもとに，不均一環境での ACO の並列化におけるボトルネックについて考察したうえで，システム全体で同期的に情報共有する処理を削減することで実時間性能の向上をねらう非同期アントコロニーシステム（Asynchronous Ant Colony System, AACS）を提案する．

## 2. アントコロニー最適化

### 2.1 巡回セールスマン問題

巡回セールスマン問題（Traveling Salesman Problems, TSP）は複数の都市と各都市間の距離（コスト）が与えられたとき，それぞれの都市を 1 度ずつ訪問し最初の都市に戻る閉路（ハミルトン閉路）のうち総距離が最小の巡回路を発見する問題である．ただしどの二都市間の距離も所与であるとする．

TSP の入力は，以下に示す重み付き完全グラフ $G$ として与えられる．

$G = (N, E)$：重み付き完全グラフ
$N = \{1, 2, \cdots, n\}$：都市
$E = \{(i, j) | i, j \in N\}$：都市間の辺（経路）
$D = \{d_{ij} | (i, j) \in E\}$：都市間の距離

TSP の出力は，ハミルトン閉路の巡回路長を最小化する最適巡回路 $s^*$ とその巡回路長 $f(s^*)$ である．ただし，$s$ はハミルトン閉路であり，$s(k) \in N$ は $s$ の $k$ 番目の都市である．本論文では，都市間の距離が対称，すなわち任意の $i$，$j$ に対して $d_{ij} = d_{ji}$ が成り立つ場合についてのみ議論する．

### 2.2 アントコロニーシステム

本研究で注目する ACO は，Dorigo ら [3], [5] によって提案された，組合せ最適化問題に対するメタヒューリスティックである．アルゴリズムは蟻の採餌行動を模したものである．蟻の集団においては，(1) それぞれの蟻が揮発性のフェロモンを通過経路上に残し，(2) 各蟻がフェロモンが多く残存する経路を選択しやすくなることによって，より餌を多く得られるような経路が形成される．ACO では，蟻の経路選択を都市間移動，フェロモンを各都市対に対するスコアの行列（フェロモン行列）としてモデル化する．そこに都市間移動の確率的選択とフェロモンの蒸発という 2 つの機構を導入することによって，有望な経路の集中的な探索と同時に探索が局所的にならないようにするのが特徴的である．

ACS は ACO の一種であり，その特徴はバランスのとれた探索の集中化と分散化にある．これを実現しているのは大域更新と局所更新の 2 段階で行われるフェロモン行列の更新である．大域更新によりそれまでに発見した最良解に基づきフェロモン行列を更新し，エージェント群全体の探索領域を最良解付近に集中させる．そして，局所更新により探索領域に揺らぎを持たせることで探索が局所に集中しにくくしている．手順は以下のとおりである．

1) 候補リストの作成

候補リストとはすべての都市についてその都市から近い位置にある都市を近い順に並べたリストである．何番目の候補まで記憶するか決定するパラメータ $cl$ は利用者の任意である．

2) 最近傍コストの計算

最近傍コスト $L_{nn}$ は最も近い位置にある都市を選択していくことでできる巡回路の長さであり，フェロモン行列の初期値を決定する際に用いられる．ランダムに初期都市を選択し，まだ訪れていない都市の中で最も近い都市に移動することを，すべての都市を訪問するまで反復することで計算される．

3) フェロモン行列の初期化

フェロモン行列 $\tau$ はすべての経路に対して次式により定義される．

$$\tau = \{\tau_{ij} | (i,j) \in E\}$$

$\tau_{ij}$ は都市 $i$ と都市 $j$ 間の辺上のフェロモンを表し，その初期値 $\tau_0$ は次式により与えられる．

$$\tau_0 = (n \times L_{nn})^{-1}$$

4) 移動する都市の選択

各エージェントは以下の手順で次の都市を選択する．

(i) $q \in [0,1]$ を一様乱数とするとき，$q \leq q_0$ の場合は手順 (ii) を行い，$q > q_0$ の場合は手順 (iii) を行う．ここで，$q_0$ は利用者によって任意に設定されるパラメータである．

(ii) 候補リストの中に未訪問都市がある場合はそれらの都市について，ない場合はそれ以外の未訪問都市について，次式をもとに評価値を計算し，評価値が最大の都市を次の都市として選択する．ここで，$r$ は現在の都市，$s$ は次の都市，$\beta$ は都市間の距離をどの程度重視するかを表す正のパラメータである．

$$p^*_{rs} = \tau_{rs} \cdot [\eta_{rs}]^\beta$$
$$\eta_{rs} = \frac{1}{d_{rs}}$$

(iii) 候補リストの中に未訪問都市がある場合は，それらの都市の集合 $N'$ の各要素に対して次式を用いて移動確率を計算し，次の都市を確率的に選択する．候補リストの中に未訪問都市がない場合は，それ以外の未訪問都市に対して手順 (ii) と同様に評価値を計算し，評価値が最大の都市を次の都市として選択する．

$$p_{rs} = \frac{\tau_{rs} \cdot [\eta_{rs}]^\beta}{\sum_{u \in N'} \tau_{ru} \cdot [\eta_{ru}]^\beta}$$

5) フェロモン行列の局所更新

各エージェントが次の都市を選択した際，選択した次の都市までの経路に対して次式に基づいてフェロモン行列の局所更新を行う．$\rho$ は蒸発率であり，$0 < \rho < 1$ である．この式はエージェントが次の都市を選択するたびに適用される．

$$\tau_{rs} \leftarrow (1-\rho)\tau_{rs} + \rho \cdot \tau_0 \quad (1)$$

6) フェロモン行列の大域更新

すべてのエージェントが巡回路を生成したとき，各巡回路の総距離を求める．その中で最小の巡回路長がこれまでに見つかった最良巡回路長よりも小さい場合は，その巡回路をその時点での最良巡回路として登録する．そして，その最良巡回路上のフェロモンを以下の式に基づいて大域更新する．ここで，$s$ は最良巡回路，$L$ はその長さである．$\psi$ は $0 < \psi < 1$ を満たす定数である．

$$\tau_{rs} \leftarrow (1-\psi)\tau_{rs} + \psi \cdot \frac{1}{L} \quad (2)$$

7) 3-opt 法の適用

3-opt 法は TSP に対してよく用いられるヒューリスティックであり，各エージェントが生成した巡回路を改善する目的で ACS に導入されている．3-opt 法では，巡回路に対して適当な 3 本の辺を選び辺の先を交換し，交換前と交換後の巡回路長を比較し巡回路が改善されていれば交換後の巡回路を新たな解候補とする [10]．

8) 4) 以降を，終了条件が成立するまで繰り返す

### 2.3 アントコロニーシステムの並列化

ACO アルゴリズムの並列化に関する研究は活発に行われており，ここで特に注目する ACS 並列化に関しては，フェロモン行列をどのようにしてシステム内で共有するかという点に各方式の特徴が表れる．これには大きく分けて，1 つのプロセッサにつき 1 つのコロニーを割り当て，複数のコロニーを並列に実行する並列アントコロニー方式と，1 つのプロセッサにつき 1 匹のエージェントを割り当て，複数のエージェントが並列に探索を行う並列アント方式とがある．

並列アントコロニー方式の例としては，Lv らの手法 [11] や Sameh らの手法 [14] があげられる．前者ではフェロモン行列をすべてのコロニーの間で共有し，したがってフェロモン行列へのアクセスの排他制御によりエージェント間での待ちが発生する．後者では各コロニーがそれぞれにフェロモン行列を持ち，一定の間隔でフェロモン行列をシステム全体で共有するための通信を同期的に行う．アクセスのたびに待ちが発生することは避けられるが，依然として上述のようなシステム全体での同期的共有処理が一定期間ごとに必要であり，そのオーバヘッドと収束時間とのトレードオフが指摘されている．

他方，並列アント方式の事例としては，Randall らによる

並列アントコロニーシステム（Parallel ACS, PACS）[13]がある．PACSではプロセッサはmaster-slave型の構造をとり，masterはフェロモン行列の更新とslaveの終了を行い，slaveは1匹の蟻として問題領域の探索を行う．

masterはslave全体に対してフェロモン行列をブロードキャストし，また，各slaveに対して初期都市を与える．slaveはそれをもとに巡回路の探索を開始する．slaveは次都市を決定するごとにmasterに対してそれを通知し，masterはそれをもとにフェロモン行列を局所更新してブロードキャストする．slaveが巡回路の生成を終えると，masterにおいて巡回路とそのコストが集約され，最良の巡回路をもとにフェロモン行列の大域更新とブロードキャストが行われる．これを，終了条件が成立するまで繰り返す．

Randallらは MPIを用いてPACSを実装し，都市数の違う8種類のTSPを対象に評価実験を行っている．その結果TSPの都市数が300以上の場合は順次型よりも実行時間が短くなることを示している．また，プロセッサを2〜8個用いた実験でプロセッサを増やすにつれて実行時間が線形に減少することを示している．しかし全体での同期点が多数存在するため，フェロモン行列共有のための待ち時間が発生し早く処理が終わった場合でも次の探索をすみやかに行うことができないという問題点が残る．

ACOの最大の特徴は各エージェントの探索による知見をフェロモンという形で蓄積し，有望な解の周辺に探索を集中させる点にある．したがって，並列化するうえでプロセッサ間でのフェロモンの共有は不可欠である．しかし，いずれの手法もフェロモン行列共有の過程で待ち時間が発生している．この待ち時間の削減がACS並列化のうえできわめて重大な課題となっており，ハードウェア化による高速化なども提案されている [23]．

## 3. 非同期アントコロニーシステム

前節で述べた，各種の並列ACS共通の問題点の分析に基づき，本研究では非同期アントコロニーシステム（Asynchronous ACS, AACS）を提案する．AACSは上述のPACSを，以下の2点について変更したものである．

- フェロモン行列を各プロセッサごとに保有する．
- 全体でいっせいにフェロモン行列の共有処理を行うことをやめ，更新情報を受け取る側の適当なタイミングで個別にフェロモン行列を受け取るようにする．

前者により，排他制御による待ち時間を削減する．排他制御自体を行う必要がなくなり，待ち時間も発生しなくなると考えられる．後者はフェロモン行列の共有処理における待ち時間の削減が目的である．更新情報を受け取る側すべてが同じタイミングで共有処理に到達すれば待ち時間は発生しないが，本研究で想定する不均一な計算環境ではそのような状況は起こりにくいと考えられ，個別に共有処理をするよう改めることで待ち時間を削減する．ただし，この戦略を用いることでシステム内におけるフェロモン行列のコピーの間の一貫性は失われる．そのため一見すれば最適解の発見が保証されないことが懸念される．これに対して本論文では，Stutzleらの証明 [15], [20] を準用することで，この戦略を用いた場合の最適解発見を保証できることを示す．

### 3.1 アルゴリズム

付録A.1にAACSにおけるmasterの処理を示す．masterの役割はフェロモン行列の更新とslaveの制御である．

AACSの開始時にmasterは各種パラメータの初期化を行った後，slaveからの通信要求の受付を開始する．

masterはslaveから次の都市を通知されたときあるいはすべてのslaveが巡回路を生成したとき，フェロモン行列を更新規則に基づき更新する．slaveから次の都市への移動を通知された場合は，その都市とslaveが現在いる都市の間の辺を局所更新し，その内容を全slaveへと通知する．この通知は各slaveの持つバッファに蓄積される．

一方，slaveの動作ステップ数が都市数と同じになったとき，すなわち巡回路を生成し終えた場合は，masterはその巡回路とコストとをslaveから受け取って保持したうえで，そのslaveに対して新たな初期都市と，その時点での最新のフェロモン行列を与え，次の巡回路生成を開始させる．すべてのslaveが1つ以上の巡回路の生成を完了した時点で，それまでに受け取った全巡回路の情報を用いてフェロモン行列の大域更新を行う．この更新情報がslaveのバッファに送られるのは局所更新と同様である．

また，付録A.2にslaveの処理を示す．slaveの役割は問題領域の探索である．

slaveは，探索の第1ステップにおいてはランダムに初期都市を選択してmasterにそれを通知する．それ以外のステップにおいては，フェロモン行列を用いて問題領域の探索を行い，次の都市を選択するとそれをmasterに通知する．次にmasterからのフェロモン更新情報が置かれるバッファを確認し，新しい情報があればそれをもとにフェロモン行列を更新し，そのフェロモン行列を用いて次の探索を行う．新しい情報がない場合は更新を行わず，すでに取得済みのフェロモン行列を用いてすぐに次の探索を行う．

masterからslaveへの通信はバッファを介した非同期通信となっており，また，探索を終了したslaveは他のslaveの探索終了を待つことなく次の経路探索を開始することができる．その一方で，slaveが探索の最中に用いるフェロモン行列は，大域的に見て最新のものである保証はなく，また一般には，任意の瞬間においてslaveどうしは異なる情報に基づいて探索を行うこととなる．

### 3.2 収束証明

ACSにおけるフェロモンの収束と最適解発見の保証は

Sutzle ら [15] により証明されている．ここではそれらの証明を AACS に準用することで，AACS におけるフェロモンが収束し，そのとき最適解が発見されることを示す．

まず，フェロモン量の上限と下限に関する以下の補題を示す．

**補題 1** いかなる繰返しにおいても，どの辺 $(i,j) \in E$ 上のフェロモンにも下限 $\tau_{min}$ と上限 $\tau_{max}$ が存在する．

$$\tau_{min} \leq \tau_{ij}(m) \leq \tau_{max}$$

ただし，$\tau_{ij}(m)$ は繰返し $m$ 回目における辺 $(i,j)$ 上のフェロモンである．

(証明)

まず ACS において補題 1 が成り立つことを証明する．式 (1)，式 (2) より局所更新の式と大域更新の式はどちらも次式のように同じ漸化式で表現することができる．ここで，$a_n$ は $\tau_{ij}(m)$ に対応し，$b$ は $\tau_0, g(s)$ に $\phi$ は $\rho$，$\psi$ に対応する．

$$a_{n+1} = (1-\phi)\cdot a_n + \phi \cdot b \qquad (m \geq 0)$$

この数列の一般項は，

$$a_m = (a_0 - b)(1-\phi)^m + b$$

である．

$m$ について微分し導関数を求めると

$$a'_m = (a_0 - b)(1-\phi)^m ln(1-\phi)$$

となる．$(1-\phi)^m > 0$，$ln(1-\phi) < 0$ であるから，$a_0$ と $b$ の値の大小のみを見ればよい．

$a_0 < b$ のとき，$a'_m > 0$ となり，$a_0$ が最小の値をとり $b$ が最大の値をとるのは，最適巡回路 $s^*$ 上のフェロモンが初期値の辺に対して大域更新のみを繰り返した場合である．このとき $a_0 = \tau_0$，$b = g(s^*)$ となる．

また $a_0 > b$ のとき，$a'_m < 0$ となり，$a_0$ が最大の値をとり $b$ が最小の値をとるのは，最適巡回路 $s^*$ 上のフェロモンが $g(s^*)$ の辺に対して局所更新のみを繰り返した場合である．このとき $a_0 = g(s^*)$，$b = \tau_0$ となる．

以上により，ACS においていかなる繰返しにおいてもどの辺上のフェロモンにも最小値 $\tau_{min}$（$= \tau_0$）と最大値 $\tau_{max}$（$= g(s^*)$）が存在する．

一方 AACS においてフェロモン行列は master と各 slave が 1 つずつ保持しているが，更新が行われるのは master のフェロモン行列のみである．その際のフェロモンの更新ルールは ACS 同様，式 (1) と式 (2) を用いるため AACS にも上記の証明はそのまま成立する．したがって，AACS においても補題 1 は成り立つ．

□

補題 1 が成り立つことから，Zhao ら [20] により与えられた最適解発見を保証する以下の定理が AACS においても成り立つ．

**定理 1** $P^*(m)$ を $m$ 回目の繰返しで最適巡回路が少なくとも 1 回発見される確率とすると，任意の小さな数 $\epsilon > 0$ に対して以下が成り立つ．

$$P^*(m) \geq 1 - \epsilon$$

また，$m$ を限りなく大きくするとき，以下が成り立つ．

$$\lim_{m \to \infty} P^*(m) = 1$$

定理 1 の証明はフェロモンに上限と下限が存在するとき最適解が発見される確率には下限が存在し，繰返し回数を限りなく大きくするとその下限が 1 に近づくということを用いて示されている．これは AACS においてもそのまま成立する．

以上により，探索に古いフェロモン情報を用いる AACS においても最終的には最適解が発見されると結論づけることができる．

## 4. 評価実験

AACS の PACS との大きな違いは，各エージェントが計算に用いるフェロモン行列がシステム全体で同じであるか否かである．

PACS ではすべてのフェロモン行列は一貫性が保たれておりつねに最新であるが，その反面同期的にフェロモン行列の共有を行うための待ち時間が発生している．一方 AACS では master のフェロモン行列に更新があった場合，更新情報をいったん各 slave のバッファに書き込み slave は処理の合間の適当なタイミングでその情報を読み込む．そのため各 slave 間のフェロモンに一貫性はなく，最新のフェロモン行列を用いて探索する slave もいれば，少し古いフェロモン行列を用いて探索する slave もいる．しかし，PACS におけるようなフェロモン行列共有のための待ち時間は発生しない．この違いにより，AACS は PACS と比較して解の収束に要する繰返し回数が増加する可能性がある．しかし，繰返しあたりの実行時間は AACS のほうがフェロモン行列共有のための待ち時間を削減した分短くなると想定される．

これらの予想から，AACS は同じ質の解を得るために PACS よりも繰返し回数が必要になるものの，実時間での比較では性能が向上する可能性がある．この予想に基づいて以下のような評価実験を行った．

### 4.1 実験環境

本研究では TSPLIB[*1] のベンチマーク問題を用いる．**表 1** に評価実験で用いたベンチマークとその都市数，最適巡回路のコストを示す．TSP や ACO，ACS の応用における問題のサイズは様々なものがあるが，序論で述べたようなシミュレーションに基づく設計への応用では，たとえば

[*1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

表 1 評価実験で用いたベンチマーク問題
**Table 1** Benchmark used in experiments.

| 問題名 | 都市数 | 最適巡回路のコスト |
|---|---|---|
| eil51 | 51 | 426 |
| kroA100 | 100 | 21,282 |
| att532 | 532 | 27,686 |
| pcb1173 | 1,173 | 56,892 |
| d1655 | 1,655 | 62,128 |
| u2152 | 2,152 | 64,253 |

表 2 パラメータの値
**Table 2** Parameters used in experiments.

| パラメータ | 値 |
|---|---|
| $\beta$ | 2 |
| $\rho$ | 0.1 |
| $\psi$ | 0.1 |
| $q_0$ | 0.9 |
| $cl$ | 20 |

自動車における車車間ネットワークに関する研究開発の事例 [7], [17] では数百から 1,000 ノード程度が想定されている．また，配送計画問題への応用 [2], [19] においても数百程度までの大きさが想定されている．以上のことから想定する問題の規模としてこれらを選択した．

その一方，この程度の大きさの問題に大規模計算環境が必要かという議論はありうる．実際，表 1 に示した問題は TSPLIB の中では問題の規模としては比較的小さな部類に属しており，TSPLIB で最大のものは都市数 85,900 で，最適巡回路長は 142,382,641 に達する．本研究の立場として，比較的小さな問題ではあっても，多量の試行を並行して行える点で，大規模計算環境を用いることのメリットはあると考えている．

実験に用いたシステムは，名古屋大学情報基盤センターに設置されたスーパーコンピュータ HX600 である．各ノードは 4 つの Opteron 8380（2.5 GHz，クアッドコア）からなり，ノードあたりの演算性能は 160 GFLOPS である．

AACS のプログラムの実装は C++と MPI を用いて行った．プログラム中で使用する各パラメータは**表 2** に示す値を用いた．また，実験 1 で用いる PACS のアルゴリズムは Randall らにより提案されているアルゴリズムをもとに AACS と同様の条件で実装した．

### 4.2 実験 1：並列アントコロニーシステムとの比較

エージェント群全体が巡回路を生成するのべ回数（以下「巡回路生成回数」と呼ぶ）の最大値（以下「最大巡回路生成回数」と呼ぶ）を固定して各アルゴリズムを実行したときの実行時間と解の質を比較するために評価実験を行った．表 1 の TSP に対して，最大巡回路生成回数を 10,000，slave の数を 10 として各プログラムを実行するという試行を 100 回繰り返し，平均実行時間，フェロモン行列の送受信に要した時間（以下「情報共有時間」と呼ぶ），および平均誤差率の 3 項目を調べた．1 試行の終了条件は巡回路生成回数が最大巡回路生成回数に達することである．

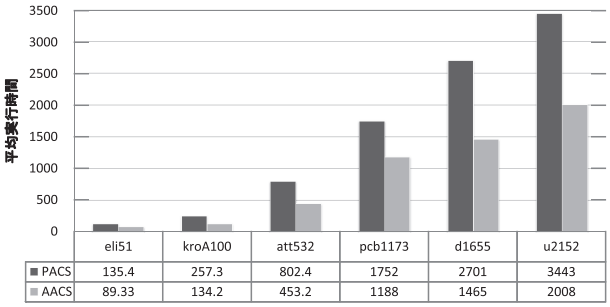


図 1 平均実行時間（単位：秒）
**Fig. 1** Mean time of calculation.

PACS においては，局所更新で最初の slave がフェロモン更新要求を master に出してから更新されたフェロモン行列を全 slave が受け取り終えるまでの時間と，大域更新で最初の slave が巡回路のコストを master に通知してから更新されたフェロモン行列を全 slave が受け取り終えるまでの時間とを計測し，それらの総和からフェロモン行列の更新のための master における計算時間を引いたものをもって情報共有時間とした．

また，AACS においては，局所更新で各 slave が master にフェロモン更新要求をわたす時間，大域更新で各 slave が master に巡回路をわたす時間，および master の持つフェロモン行列を各 slave が受け取る時間の総和を情報共有時間とした．

誤差率は，表 1 に示した最適巡回路のコストを用い，以下の式により求めた．

$$\text{誤差率} = \frac{\text{求めた巡回路のコスト} - \text{最適巡回路のコスト}}{\text{最適巡回路のコスト}}$$

**図 1** に平均実行時間を示す．都市数の増加につれて実行時間が増大し，どの問題についても AACS の実行時間が PACS の実行時間よりも短かった．都市数の小さい `eli51` および `kroA100` においては最適解が各試行のごく初期の段階（1 秒未満）で発見された．一方，それ以外の都市数 500 の問題では最適解まで到達していないが，これは主に最大巡回路生成回数の不足が原因と考えられる．

**図 2** に平均情報共有時間を示す．本実験においては，AACS と PACS の実行時間の差の原因はおもに情報共有時間の差に求められることが分かるが，それらは厳密には一致しない．この要因としては，PACS と AACS とで情報共有時間の定義に差異があることや，通信手順に関する実装上の違いなどが影響している．

**図 3** に試行の終了時点での誤差率の平均を示す．前述のように `eli51` および `kroA100` では最適解が求まったため誤差率は 0 である．一方，500 都市以上の問題では最適解

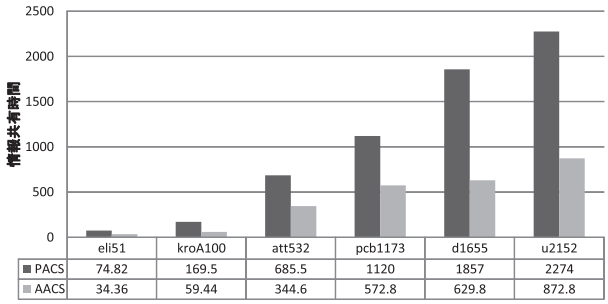


**図 2** 平均情報共有時間（単位：秒）
**Fig. 2** Experiment 1: Mean time for sharing (seconds).

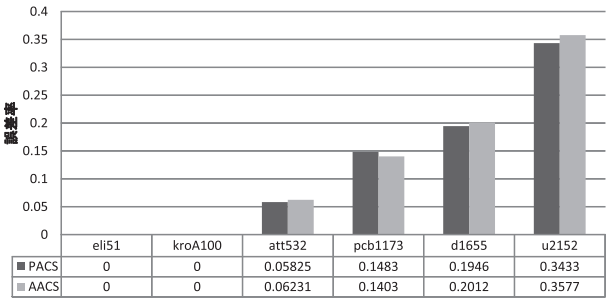


**図 3** 実験 1：平均誤差率
**Fig. 3** Experiment 1: Mean errors.

**表 3** 実験 1：誤差の標準偏差
**Table 3** Experiment 1: Standard deviation of errors.

| 問題 | PACS | AACS |
|---|---|---|
| att532 | 0.105 | 0.098 |
| pcb1173 | 0.134 | 0.142 |
| d1655 | 0.121 | 0.148 |
| u2152 | 0.212 | 0.201 |

**表 4** 実験 1：誤差率 0.5 以下となるまでの平均時間（単位：秒）
**Table 4** Mean time to reduce errors under 0.5 (seconds).

| 問題 | PACS | AACS |
|---|---|---|
| att532 | 128.2 | 79.8 |
| pcb1173 | 1028.3 | 328.9 |
| d1655 | 983.4 | 608.3 |
| u2152 | 2532.3 | 1482.9 |

が必ずしも求まらなかったため，最大巡回路生成回数に達した時点で誤差が生じた．誤差が生じた問題に関して，誤差の標準偏差を**表 3** に，誤差率が 0.5（得られた解のコストが，最適解のコストの 1.5 倍）以下となるまでの平均時間を**表 4** に示す．問題のサイズに応じて標準偏差が大きくなり，一定の解の精度が得られるまでの時間は長くなっているが，PACS と AACS の間での顕著な傾向の違いは見られなかった．

### 4.3 実験 2：1 回の共有あたりに更新される要素数

PACS において master のフェロモン行列に更新が行われた場合，全 slave はただちにフェロモン行列の共有処理を行い自身のフェロモン行列を最新のものにする．一方 AACS ではこのようなシステム全体でのフェロモン行列の共有を行わないため，slave の持つフェロモン行列は必ずしも最新のものではない．これにより，同じ回数の巡回路生成を行った場合には，PACS と比較して AACS の誤差率は悪化するのが自然と考えられる．しかしながら，図 3 に示したように，実験 1 においては大きな誤差率の差が見られなかった．

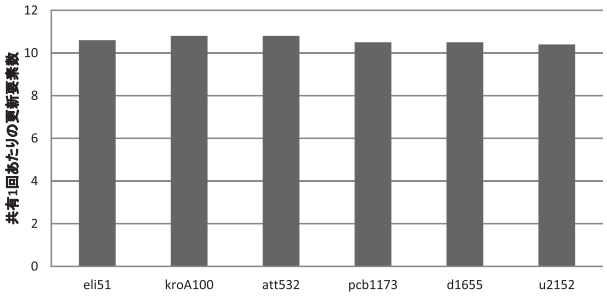


**図 4** 実験 2：共有 1 回あたりの更新要素数
**Fig. 4** Experiment 2: Number of updated elements per information sharing.

この現象を分析するため，master と slave のフェロモン行列の間の差異について実験によって調べた．実験は表 1 の TSP に対して slave の数を 10 として AACS を実行し，slave のフェロモン行列において 1 回の共有あたりに更新が行われた要素数を調べることで行った．**図 4** に 1 回の共有あたりに更新が行われた要素数を示す．

slave のフェロモン行列において 1 回の共有処理あたりに更新が行われたフェロモン行列の要素数，すなわち，PACS のように毎度の更新が反映されていないためにいわば古い情報となってしまった部分である．一方で，フェロモン行列全体の要素数は都市数の二乗に比例しており，フェロモン行列全体に対しては非常に小さい範囲にとどまっている．このため，実験 1 で AACS の誤差率が PACS に対してさほど悪化しなかったと考えられる．

しかしながら，この状況はつねに生ずるわけではない．AACS において，大域更新は全部の slave が巡回路を 1 つ以上生成し終わらないかぎり行われないという性質が問題となりうる．

slave 間の処理能力に大きなへだたりがある場合を考える．たまたま非常に遅い slave が他の slave よりも非常に良い解を選んでいた場合を考えると，遅い slave が算出する良い解に関する情報がシステム全体に伝搬されず，速い slave がいわば見込みの薄いところを大量に探索するということが起きえ，この場合は生成巡回路数に対して精度が向上しないはずである．また，遅い slave にとっては共有処理ごとにフェロモン行列が大きく書きかわることになるため，実験 2 で見たような状況があてはまらないと想定される．他にも，極端に遅い slave が少数ある場合，あるい

は逆に極端に速い slave が少数ある場合など，slave 間の速度分布によって差異があると考えられる．

また，slave 数の大きさも影響する可能性がある．slave 数が増えれば更新の回数もそれにつれて増えるため，実験 2 で見たような共有処理時の更新量は増大し，誤差率への影響が予想される．

本論文の範囲ではこれらの状況すべてを考慮した本格的な検証を行うに至っていないが，上に述べた処理能力のへだたりおよび分布と性能との関係については精査を行う必要がある．各 slave に待ち時間をランダムに与えることによって処理能力のばらつきを模した初期的な実験を行っている．処理能力のばらつきによって誤差率に若干の上昇は認められるものの，上の考察で述べたような slave 数や速度の分布と誤差率との明確な関係はいまだ判明していない．

## 5. 結論

本論文では，メタヒューリスティクスの一種である ACS の並列化に際し，フェロモン行列の共有処理をシステム全体でいっせいには行わない非同期アントコロニーシステムの提案を行った．これにより，システム内で異なるフェロモン行列を用いた探索が同時に行われることになるが，その状況においても従来の同期を行う方式と同様に最適解への収束性が保証されることを示した．またこれによって，プロセッサ間の待ちが少なくなり，最適解の探索に要する時間が削減されうることを実験により示した．

本提案手法の性能を左右する要因としては，4 章でも述べたような slave 間の処理速度のばらつきや，slave の数，適用する問題の規模や性質など，様々なものをあげることができる．それらに対する本手法の定性的・定量的性質は必ずしも明らかになっていない．今後，さらに多数の事例に対する実験などを通じた性能の評価を進めていく必要がある．

本手法によって，並列に動作するエージェントの数が増え，相互の待ちが発生することによる計算効率低下は抑えられると考えられるが，その一方で，単純にエージェントの数だけを増やしても探索の性能は必ずしも向上しないことが一般に知られている．これは，エージェントの数の増加にともなって探索領域の集中化が十分に行われず解の収束がかえって遅くなってしまうからであると考えられる．我々が別途行った実験においても，AACS でベンチマーク中の 532 都市の TSP を解くために最適なエージェントの数は 12 であった．今後の課題として，今回行った並列アント方式の非同期化だけでなく，並列アントコロニー方式と組み合わせることにより，適切な規模のコロニーを大量に用いる方式をあわせて検討していく必要がある．

また，上述のように単一の master が複数の slave を制御する方式では，非同期通信を行ったとしても master となるプロセッサの処理能力が全体のボトルネックとなりうることが明らかであり，この部分の並列化ないし分散化も今後本手法の大規模な問題への適用のうえでは必要となる．

**謝辞** 本研究の成果の一部は，戦略的情報通信研究開発推進制度（SCOPE:091603006）および科研費（23500175）の支援を受けている．

### 参考文献

[1] Belding, T.C.: The Distributed Genetic Algorithm Revisited (1995).
[2] Bell, J.E. and Griffis, S.E.: Swarm Intelligence: Application of the Ant Colony Optimization Algorithm to Logistics-Oriented Vehicle Routing Problems, *Journal of Business Logistics*, Vol.31, No.2, pp.157–175 (online), DOI: 10.1002/j.2158-1592.2010.tb00146.x (2010).
[3] Dorigo, M. and Caro, G.D.: Ant Algorithms for Discrete Optimization, *Artificial Life*, Vol.5, No.2, pp.137–172 (1999).
[4] Dorigo, M. and Gambardella, L.M.: Ant Colony System:A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Trans. Evolutionary Computation*, Vol.1, No.1, pp.53–66 (1997).
[5] Dorigo, M. and Stutzle, T.: *Ant Colony Optimization*, MIT Press (2004).
[6] Endo, T. and Matsuoka, S.: Massive supercomputing coping with heterogeneity of modern accelerators, *Proc. IEEE International Symposium on Parallel and Distributed Processing* (*IPDPS2008*), pp.1–10 (2008).
[7] Gunes, M., Sorges, U. and Bouazizi, I.: ARA-the ant-colony based routing algorithm for MANETs, *Proc. International Conference on Parallel Processing Workshops*, pp.79–85 (online). DOI: 10.1109/ICPPW.2002.1039715 (2002).
[8] Holland, J.: *Adaptation in Natural and Artificial Systems*, University of Michigan Press (1975).
[9] Kirkpatrick, S., Gelatt, Jr., C.D. and Vecchi, M.P.: Optimization by Simulated Annealing, *IBM Research Report*, RC 9355 (1982).
[10] Lin, S.: Computer solutions of the traveling salesman problem, *Bell System Technical Journal*, Vol.44, pp.2245–2269 (1965).
[11] Lv, Q. and Xia, X.: A Parallel ACO Approach Based on One Pheromone Matrix, *Proc. 5th Int. Workshop on Ant Colony Optimization and Swarm Intelligence*, pp.332–339 (2006).
[12] Mülenbein, H.: Evolution in time and space – the parallel genetic algorithm, pp.316–337 (1991).
[13] Randall, M. and Lewis, A.: A Parallel Implementation of Ant Colony Optimization, *Journalof Parallel and Distributed Computing*, Vol.62, No.9, pp.1421–1432 (2002).
[14] Sameh, A. and Ayman, A.: Parallel Ant Colony Optimization, *International Journal of Reseach and Reviews in Computer Science*, Vol.1, No.2 (2010).
[15] Stutzle, T. and Dorigo, M.: A short convergence proof for a class of ant colony optimization algorithm, *IEEE Trans. Evolutionary Computation*, Vol.6, No.4, pp.358–365 (2002).
[16] Suzuki, M.: Evolutionary Acquisition of Complex Behaviors through Intelligent Composite Motion Control, *Proc. 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation* (2005).
[17] Wang, J., Osagie, E., Thulasiraman, P. and Thulasiram, R.K.: HOPNET: A hybrid ant colony optimization

routing algorithm for mobile ad hoc network, *Ad Hoc Networks*, Vol.7, No.4, pp.690–705 (online), DOI: 10.1016/j.adhoc.2008.06.001 (2009).
[18] Wellman, M.: A Market-oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems, *Journal of Artificial Intelligence*, Vol.1, pp.1–23 (1993).
[19] Yu, B., Yang, Z.-Z. and Yao, B.: An improved ant colony optimization for vehicle routing problem, *European Journal of Operational Research*, Vol.196, No.1, pp.171–176 (online), DOI: 10.1016/j.ejor.2008.02.028 (2009).
[20] Zhao, B. and Li, S.: A Convergence Proof for Ant Colony Algorithm, *Proc. 6th World Congress on Intelligent Control and Automation*, pp.3072–3075 (2006).
[21] 上田一康，石井克哉，永井　亨，鈴木大介，熊澤 峰夫：遺伝的アルゴリズムを用いた弾性波動場のパラメータ推定，日本応用数理学会年会講演予稿集，pp.347–348 (2010).
[22] 相吉英太郎，安田敬一郎：メタヒューリスティクスと応用，電気学会 (2007).
[23] 石山直幸，吉川雅弥，寺井秀一：アントコロニー最適化手法の専用ハードウェアの設計と評価, 電子情報通信学会技術研究報告 AI，人工知能と知識処理，Vol.108, No.119, pp.1–6 (2008).
[24] 三井和夫，大崎　純，大森博司，田川　浩，本間俊雄：発見的最適化手法による構造のフォルムとシステム，コロナ社 (2004).

# 付　　録

## A.1 AACS の master のアルゴリズム

**Input**:tsp file,parameters
**Output**:*globally_best_tour*,*globally_best_cost*

Initialize parameters and distance matrix
Calculate *nearest_neighbor_cost*
Initialize $\tau_0$ with *nearest_neighbor_cost*,and broadcast it
Initialize the pheromone matrix with $\tau_0$
**while**(termination criterion is not satisfied)
  **for**(*slave_id* = 1 to the number of slaves)
    **if**(there is a communication request from *slave_id*)
      **if**(*step*[*slave_id*] == 1)
        Recieve *initial_city* from the slave *slave_id*
      **else**
        Recieve *next_city* from the slave *slave_id*
        Local update the pheromone of the edge
        Put local update information on all slave buffer
      **end if**
      **if**(*step*[*slave_id*] == the number of cities)
        Recieve *tour* and *cost* from the slave *slave_id*
        Push *tour* into tour queue of *slave_id*
        Push *cost* into cost queue of *slave_id*
        **if**(all of cost queue size $\geq$ 1)
          **for**(*slave_id* = 1 to the number of slaves)
            Pop tour queue of *slave_id*
            Pop cost queue of *slave_id*
          **end for**
          *iteration_best_cost* = the best cost in them
          **if**(*iteration_best_cost* < *globally_best_cost*)
            *globally_best_tour* = *iteration_best_tour*
          **end if**
          Global update the pheromone of the edges which belong to *globally_best_tour*
          Put global update information on all slave buffer
        **end if**
      **end if**
      **if**(it is the time to terminate the slave *slave_id*)
        *termination_flag* = true
      **end if**
      Send *termination_flag* to the slave *slave_id*
      **if**(*step*[*slave_id*] == the number of cities)
        *step*[*slave_id*] = 1
      **else**
        *step*[*slave_id*] = *step*[*slave_id*] +1
      **end if**
    **end if**
  **end for**
**end while**

## A.2 AACS の slave のアルゴリズム

Initialize parameters and distance matrix;
Recieve $\tau_0$;
Initialize the pheromone matrix with $\tau_0$;
Create buffer for update information;
**while**(*termination_flag* is false)
  **if**(*step* == 1)
    Choose *initial_city* randomly;
    Send *initial_city* to the master;
  **else**
    Choose *next_city*;
    Send *next_city* to the master;
  **end if**
  **if**(*step* == the number of cities)
    Apply Three-opt to *tour*;
    Calculate *cost* of *tour*;
    Send the tour and *cost* to the master;
  **end if**
  Check update information buffer;
  **if**(there is new update information)
    Get update information from the buffer;
  **end if**
  Recieve *termination_flag* from the master;
  **if**(*step* == the number of cities)
    *step* = 1;
  **else**
    *step* = *step* + 1;
  **end if**
**end while**

**八槇 博史** （正会員）

名古屋大学情報基盤センター准教授．1999 年京都大学大学院情報学研究科博士後期課程修了．博士（情報学）．1999 年同研究科助手．2001 年同講師．2006 年より現職．マルチエージェントシステムのネットワーク応用，情報セキュリティ，シミュレーション技術等に興味を持つ．電子情報通信学会，人工知能学会，ACM，IEEE 各会員．

**中村 悟**

2011 年名古屋大学大学院情報科学研究科博士課程前期課程修了．在学中はマルチエージェントシステムに関して研究．

**佐藤 亮介**

2011 年名古屋大学工学部電気電子情報工学科卒業．現在，同大学院情報科学研究科情報システム学専攻博士前期課程に在籍．メタヒューリスティクス，および高解像度差分法に興味を持つ．