階層型アントコロニー最適化法のマルチコア型並列計算機への実装と性能評価 ： 定期検査制約を有する車両運用計画問題への適用

北川, 幸弥
九州大学大学院工学府

辻, 康孝
九州大学大学院工学研究院

黒田, 真弘
株式会社クボタ

出版情報：TECHNICAL REPORT OF IEICE.. 113 (383), pp.51-55, 2014-01-14. THE INSTITUTE OF ELECTRONICS, INFORMATION AND COMMUNICATION ENGINEERS
バージョン：

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

階層型アントコロニー最適化法のマルチコア型並列計算機への実装と性能評価

定期検査制約を有する車両運用計画問題への適用

北川　幸弥†　辻　康孝††　黒田　真弘†††

† 九州大学大学院工学府　〒 819–0395 福岡市西区元岡 744
†† 九州大学大学院工学研究院　〒 819–0395 福岡市西区元岡 744
††† 株式会社クボタ　〒 590–0823 大阪府堺市堺区石津北町 64
E-mail: †kitagawa@zeus.mech.kyushu-u.ac.jp, ††tsuji@mech.kyushu-u.ac.jp,
†††masahiro.kuroda@nk.kubota.co.jp

あらまし　著者らは車両運用計画問題に対して複数コロニーを用いる階層型アントコロニー法（H-ACO）を提案している．H-ACO はその特徴からコロニー数の多い計算を行うと計算時間が膨大になる．そこで，H-ACO をマルチコア型並列計算機に実装する．並列計算は H-ACO プログラムの一連の流れをコロニーごとに分割し，それを並列計算機の複数スレッドに割り当てて行う．本研究では，OpenMP を用いたマルチスレッドプログラミングによる H-ACO の実装法について検討し，数値実験により計算時間の改善について評価する．
キーワード　アントコロニー最適化法，並列計算，OpenMP，スケジューリング，車両運用計画

Implementation of Hierarchical Ant Colony Optimization on Multi-core Parallel Computer and Its Performance Evaluation

Application to Rolling Stock Planning with Regular Inspection

Yukiya KITAGAWA†, Yasutaka TSUJI††, and Masahiro KURODA†††

† Graduate School of Engineering, Kyushu University, Motooka 744, Nishi–ku, Fukuoka 819–0395, Japan
†† Faculty of Engineering, Kyushu University, Motooka 744, Nishi–ku, Fukuoka 819–0395, Japan
††† Kubota Corporation, Ishizu–Kitamachi 64, Sakai-ku, Sakai-shi, Osaka 590–0823, Japan
E-mail: †kitagawa@zeus.mech.kyushu-u.ac.jp, ††tsuji@mech.kyushu-u.ac.jp,
†††masahiro.kuroda@nk.kubota.co.jp

Abstract　We developed Hierarchical Ant Colony Optimization (H-ACO) for solving railway rolling stock planning. H-ACO uses several colonies and allocates them hierarchically. However, it requires a huge amount of computation time for calculation using large number of colonies. In this paper, we implement H-ACO in parallel computation environment with multi-core processor in order to reduce its computation time. The effectiveness of the proposed method is demonstrated through a numerical experiment.
Key words　Ant Colony Optimization, Parallel Computing, OpenMP, Scheduling, Rolling Stock Planning

1. はじめに

アントコロニー最適化法（Ant Colony Optimization: ACO）は，アリのフェロモンを介した捕食行動をモデル化した進化計算の一つで，多くの組合せ最適化問題に適用されている [1]. ACO では，アリの個体群が過去の探索情報を蓄積したフェロモンを参照しながら探索を行う単位を“コロニー”と呼ぶが，ACO の解探索能力の向上させる構成法として複数コロニーで協調探索を行う Multi-colony ACO も多数提案されている [2]. 一方，実問題では問題サイズが大きいだけでなく，複雑な制約条件や評価項目が複数の場合も多い．進化計算で実問題を効率的に解くためには，(場合によっては複数の）局所探索法との組

合せが不可欠であり， Multi-colony ACO を用いた場合，計算時間の増大が避けられない．そのため Multi-colony ACO は並列計算機環境への実装が前提となる [2], [3].

著者らは，上記のような実問題として旅客鉄道の車両運用計画問題 [4] を対象とし，ACO に基づく手法の開発を行っている [5]．また探索性能の向上と多目的性を取り扱う構成法として，複数コロニーを階層的に配置した Hierarchical-ACO（H-ACO）の提案を行った [6]．そこで本研究では，H-ACO のマルチコア CPU 上への実装について検討する．並列化方式として，OpenMP を用いて各コロニー計算を複数スレッドに割り当てるコロニーレベルの並列化方式を取り，車両運用計画問題を対象とした数値実験を通して，計算時間の改善について評価する．

2. 関 連 研 究

ACO の並列化の研究は盛んに行われている [2], [3], [8], [9]. 並列化の前提として，解の探索性能を向上させるために複数のコロニーを用いた Multi-colony ACO が提案されている．複数のコロニーがそれぞれ独立した解構築を行った後，コロニー間で情報交換を行う．文献 [2] では各コロニーで得た最良解の情報共有ルール（全結合，ハイパーキューブ，リング状，独立実行）と局所探索の組み合わせが異なるモデルを並列計算機に実装し，TSP を用いて各ケースの性能比較を行っているが，各種ルール間で顕著な性能差は明らかでない．また文献 [3] は，各コロニーで得られた最良の解やフェロモン濃度マトリックスの値を交換するモデルを提案しているが，情報交換に時間がかかり交換しない場合に比べて実行時間の面で結果が悪化している．これらとは別に文献 [7] では，二つのコロニーにそれぞれ異なる目的関数とフェロモンマトリックスを与え，アリの解構築後互いの解情報を共有し 2 目的の最適化問題を解いている．

一方，単コロニーの ACO の並列化にはアリ単位の並列化手法が提案されている．文献 [8] ではマスタースレーブモデルを採用し，スレーブのアリがたどった経路の情報をマスターが受け取り，すべての経路情報をもとにフェロモンを更新している．GPGPU を用いてこのモデルをアリ単位で並列化実装し，超多スレッド並列計算を行っている．文献 [9] ではスレーブが生成した解のうち一番良い評価を受けた解の情報をもとにマスターのフェロモンを更新する．

3. Hierarchical Ant Colony Optimization

3.1 H-ACO の枠組み

著者らが提案している H-ACO [6] は，複数のコロニーを用い，コロニー間で協調しながら解探索を行うマルチコロニー型の ACO である．H-ACO では，各コロニーは独自のフェロモン情報を持ち，各コロニーはランク付けされ階層的に配置されている．ランクは目的関数のレベルに応じて割り振られる．例えば，配送問題で車両数及び移動距離（時間）の最小化では，車両数がランクを表す識別子となる．得られた解の中で最小の車両数を最上位ランクの識別子とし，一つ下位のランクには次に小さな車両数を識別子として与える．同様の方法で各コロニーに識別子を定めランク付けする．解探索の進展とともに識別子の値は更新されていく．その他，制約条件の多い問題では，制約違反の個数に応じて同様のランク付けを行うこともできる．H-ACO では，各コロニーごとにフェロモン情報を初期化し，アリの個体群による解探索を繰り返す．各反復では，全コロニーの解情報を一旦集約し，各コロニーごとにフェロモン情報を更新する．また最上位コロニーの識別子の値が更新された場合，コロニー更新を新たに導入している．なお各コロニーでの解構築とフェロモン更新は，$\mathcal{MM}$AS [10] を採用している．

3.2 解 構 築

コロニー数を Q とし，コロニーごとに m 個のアリの集団を用いて対象問題の解を反復生成する．各アリは解要素（ノード）を確率的に一つずつ選択し，解候補を生成する．h 番目のアリがノード i からノード j を選択するルールを次式に示す．

$$p_{i,j}^{h}(t) = \frac{[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}]^{\beta}}{\sum_{l \in \mathcal{N}_i^h}[\tau_{i,l}(t)]^{\alpha}[\eta_{i,l}]^{\beta}}, \quad j \in \mathcal{N}_i^h \tag{1}$$

t は反復回数，$\tau_{i,j}(t)$ は t 回目までの探索によってノード (i,j) 間に蓄積されたフェロモン情報，$\eta_{i,j}$ はノード (i,j) 間のヒューリスティック情報でノード間の長さを示す．なおこの値は対象問題ごとに定義する必要がある．$\mathcal{N}_i^h$ はアリ h がノード i から次に移動可能なノード集合を表す．α, β は二つの情報を調整するパラメータである．さらに解の改善を図るために，アリが構築した解に対して局所探索をオプションとして適用する．

3.3 解情報の集約

全コロニーの解構築後，生成した全ての解を一旦集約し，識別子の値ごとに解を分類し，各識別子ごとの最良解（iteration-best）を求める．さらにこれら Q 個の解は，各コロニーが保持している最良解 (global-best) と比較・更新を行う．

3.4 フェロモン更新

まず各コロニーにおいて，全ノード間のフェロモンを一定割合 ρ だけ揮発させ，各コロニーごとの最良解 (global-best) に含まれるノード間にフェロモンが新たに付加される．このフェロモン更新を次式に示す．

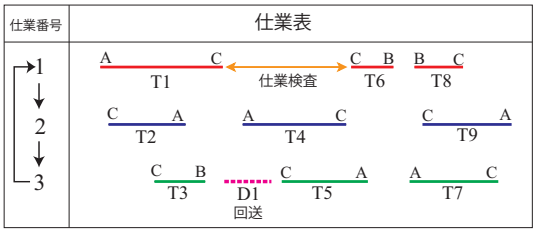
$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \Delta\tau_{i,j} \tag{2}$$

$$\Delta\tau_{i,j} = \begin{cases} \dfrac{1}{R^{bs}} & if\ (i,j) \in \boldsymbol{C}^{bs} \\ 0 & otherwise \end{cases} \tag{3}$$

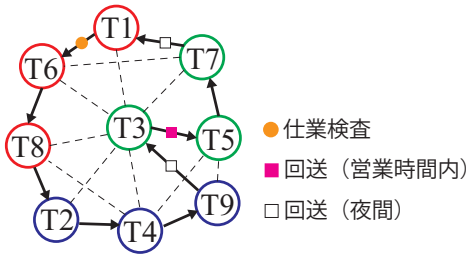
$\Delta\tau_{i,j}$ は最良解 $\boldsymbol{C}^{bs}$ の評価値 R^{bs} に従って付加されるフェロモン量である．フェロモン更新後，$\mathcal{MM}$AS のフェロモン制限によりフェロモンの過剰な集中を抑制する．この制限に関して，詳しくは文献 [10] を参照されたい．また新たに最上位コロニーの識別子の値が更新された場合には，新たに最上位コロニーを生成する．さらに各コロニーはランクを一つ下げ，最下位ランクのコロニーは破棄する．この操作をコロニー更新と呼ぶ．

4. 車両運用計画問題

車両運用計画 [4] では，与えられた列車計画に基づいて，車両に一日に担当させる列車を順番に並べた「仕業」を定める．また各車両には毎日同じ仕業を担当させるのではなく，異なる



(a) 仕業表の例



(b) 車両運用ネットワーク

図 1　車両運用計画

仕業を順番に担当させ，この一定期間の担当仕業の順番あるいは循環を「交番」と呼ぶ．この仕業と交番を定める計画を車両運用計画という．また鉄道車両は，検査設備のある駅・時間帯に仕業検査を受ける必要がある．この仕業検査は数日おきに 1, 2 時間程度かけて行われるため，運用計画の中に仕業検査計画も組み込む必要がある．運用計画は，仕業数（使用車両数）と回送数により評価されるが，これらは競合する場合もある．図 1(a) に 9 本の列車の運用計画例を示す．

運用計画の自動化について，車両運用ネットワークモデルが提案されている [12]．図 1(b) は各列車をノード，列車間のつなぎをアークとした有向ネットワークで，図 1(a) の仕業表を表現したものである．日をまたぐアークまでのノード順（例: T1 → T6 → T8）が一仕業の列車順序を示している．日またぎの他，回送や検査の設定もアークに組み込まれている．車両運用計画はこのモデルを用いて周期的な検査制約を満たしつつ最適な巡回路を求める問題に帰着され，いくつかの解法アルゴリズムが提案されている [5], [11]〜[14]．

5. H-ACO による解法とその並列化手法

5.1 H-ACO による車両運用計画アルゴリズム

まずダイヤ上の n 本の列車をノードとする車両運用ネットワークを作る．各列車ノード $i(i = 1, \cdots n)$ は，出発・到着駅，出発・到着時間を属性値として持ち，各ノード間のアークコスト $\lambda_{i,j}$ は，駅での待機時間，回送の有無，日またぎ等の接続を勘案して与える．本研究では文献 [5], [6] のように定義した．式（1）のヒューリスティック値 $\eta_{i,j}$ は $\lambda_{i,j}$ の逆数で与える．

$$\eta_{i,j} = \frac{\epsilon}{\lambda_{i,j}} \tag{4}$$

アリによる解構築では日をまたぐ列車の接続をカウントすることで，検査が必要なタイミングが分かる．そこで周期的に仕業検査を組み込むために，検査実施日のときは検査可能な接続のアークコストを小さく，実施日以外ではその値を大きくする修正を ϵ の値により行う．運用計画は仕業数（使用車両数）と回送数（または回送距離）の最小化を目的とする．アリにより生成される巡回路の評価関数は次式で与える．

$$R = 10^2 A + 10^{-1} D + 10^{-2} H + 10^{-2} D' + 10^{-3} H'/2 + P(5)$$

車両数 A，回送数 D，回送時間総和 H，夜間回送数 D'，夜間回送時間総和 H' とする．P はペナルティー項などである．

対象路線によって車両数と回送コストは競合する場合もあり，同時に最小化するのは難しい．一方，最小車両数の解と回送コストが最小の解を同時に提示できれば，意思決定者の選択肢を広げることができる．そこで多目的性を考慮した解探索を H-ACO により行う． H-ACO のランク分けの識別子に車両数を用いて Q 個のコロニーを生成及び階層化し，車両数ごとの解探索を行う．最上位コロニーにはこれまで全コロニーで生成した解の中で最小の車両数を識別子として与える．下位 $Q-1$ 個のコロニーは 3 章の方法で定め，コロニーごとに解を構築する．

5.2 局所探索

アリの解構築後， コロニーごとに局所探索をさらに適用し，車両数と回送コストの改善を図る．局所探索は車両数を削減するもの（LS1）と回送コスト（営業中: LS2 / 夜間: LS3）を削減するものを用いる．LS1 は作成された解のうち，担当列車数が少ない仕業を列車単位に分解し，他の仕業で当日中に接続できるところに接続する．LS2 は，LS1 を適用した解に対して営業時間内での回送を全て抽出し，その回送の前後で行路の部分的な繋ぎかえ，まず回送をなくす繋ぎかえを行い次に回送距離を短くする繋ぎかえを行う．LS3 は LS2 と同様の操作を夜間の回送に適用する．詳しくは文献 [6] を参照されたい．

5.3 情報集約とコロニー更新

各コロニーで生成した解の情報を一旦集約し，車両数ごとにソートする．各コロニーのフェロモンは識別子と同じ車両数でこれまでの最良の解の評価値を用いて更新する．また，識別子が更新された場合のコロニー更新の流れを図 2 に示す．

5.4 OpenMP による並列化手法

逐次計算では，解構築と局所探索は反復回数×コロニー数繰り返されるため，コロニー数が大きくなると膨大な計算時間がかかる．そこで，コロニーレベルの並列化によって各コロニー計算の処理をスレッドとして並列に実行する．並列化には OpenMP を用いる．OpenMP は指示文の挿入により簡単に並列化コードがかけるほか，対応コンパイラも多く並列計算機に容易に実装できる．実際の H-ACO の並列計算の流れを図 3 に示す．Q 個のスレッドを生成し，これらが Q 個のコロニーの処理を実行する．スレッド Q はコロニー Q の解探索を担当する．コロニーごとに独立した解構築・各局所探索後，その都度解情報をスレッド 1 に集め解をソートし，必要に応じてコロニー更新を行う．その後，フェロモン情報とコロニーごとの最良解を各スレッドに伝える．その情報をもとに各コロニーで独立したフェロモン更新を行う．以上の流れを反復回数分繰り返す．

マルチスレッド計算により，主要な解探索・局所探索パートの計算量はシングルスレッド時に比べコロニー数分の一に短縮され，計算時間を削減できる．

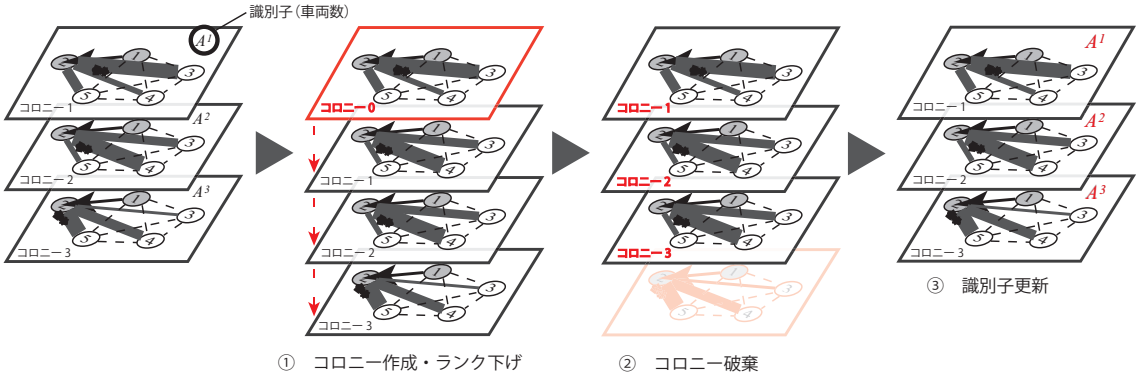


図 2　コロニー更新

5.5　同期の取り方

著者らは実際に H-ACO を並列計算機に実装したが，想定していた短縮率は達成できなかった [15]．そこで本研究ではまず，予備実験としてシングルスレッド計算とマルチスレッド計算の各パートの計算時間を比較する．各解探索パートと全スレッドで各パートが終了するまでの待ち時間，解集約・ソートの時間を計測した．例として 3 コロニー・アリ数 30 の 1 コロニーあたりの平均計算時間を図 4 に示す．実験条件は次章と同様とする．並列計算により各探索パートの計算時間は削減できた．しかし，図 3 の点線で示されている部分で解情報の同期を行っており，全コロニーの処理終了を待つ時間が反復回数分積み重なっている．特に LS2 適用後の同期のための待ち時間 W3 は他のパート後の W（0.2 ～1.0 秒）に比べて大きい．LS2 は営業時間内の回送を全て抜き出して繋ぎかえを行うため，解ごとに計算量が変化し終了時間のばらつきに原因がある．コロニー単位の並列化に加え 1 コロニーの計算を分割する新しい並列化方式を採用すると，解探索と局所探索の時間はさらに削減されるが，待ち時間は増加し並列化効率を悪化させる．

そこで，本研究ではこの W3 の解消を目的とする．LS2 の適用が一番早く終了したスレッド（コロニー）に合わせて他スレッドの LS2 適用を強制的に打ち切る．これにより，全スレッドでの LS2 適用がほぼ同時に終了し W3 を解消できる．テスト問題に本手法を適用し，効果を確認する．

6.　数 値 実 験

6.1　対象路線と実験条件

テスト問題（列車数 293，駅数 19，検査駅数 2）を用いて実験を行う．この問題は最高で 38 本の列車が同時に走っており，運用にあたって 38 本の列車が必要である．計算条件は $m = 30$・40・50，各ケースの反復回数は 500，その他の ACO パラメータは文献 [5] と同様とし，実験は Windows7 Intel Xeon(4core CPU) 2.67GHz，48GB RAM の計算機を用いた．コロニー数 1～4 のマルチスレッド計算（1 コロニーの場合はシングルスレッド）を各 5 回実行した計算時間を測定した．

6.2　実 験 結 果

まず，打ち切り処理を適用しない場合の W3 を図 5 に示す．4 コロニー，個体数 50 の計算では約 9 秒かかっており，計算時間の削減効率を悪くしている．次に，打ち切り処理を施した

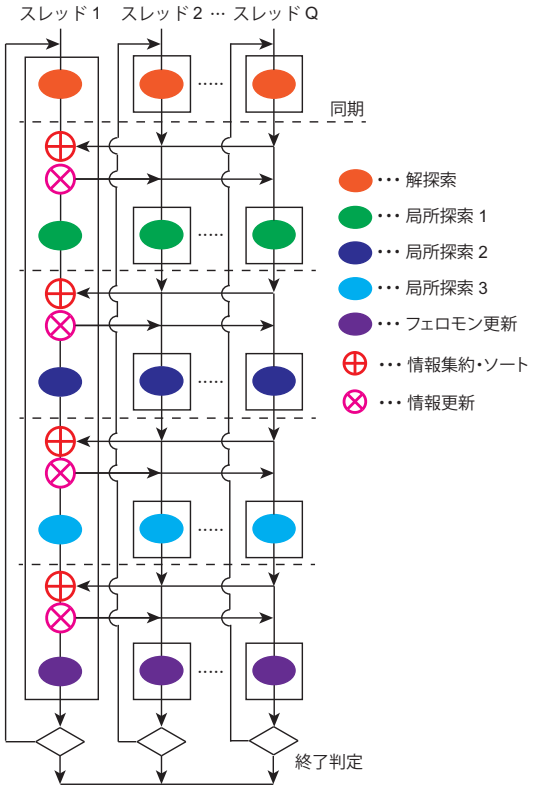


図 3　マルチスレッド計算の流れ

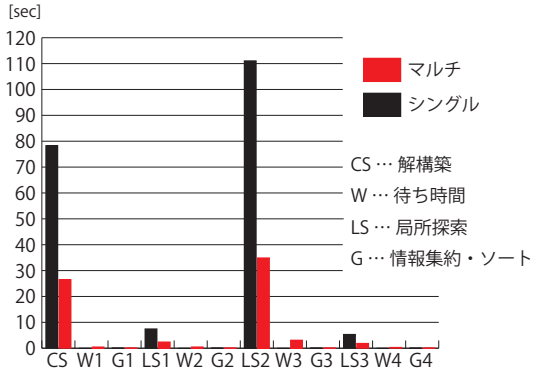


図 4　各 H-ACO パートごとの平均計算時間

場合の W3 を図 6 に示す．打ち切り処理により適用前に比べて待ち時間がほぼ解消できている．また処理を打ち切られたアリはごく少数で H-ACO の性能に大きな変化は見られなかった．打ち切りを実装したコロニー数別の計算時間を表 1 に示す．各試行で 1～2 分程度と短い時間で解を得ることができた．実際

表 1　1 試行あたりの平均計算時間 (秒)

個体数＼コロニー数	1	2	3	4
30	77.7	65.5	68.2	73.9
40	103.3	88.6	91.0	97.2
50	126.7	110.5	112.3	115.1

表 2　H-ACO の性能

コロニー数	車両数	回送数（時間 [分]）	夜間回送数（時間 [分]）
1	46.0	54.6 (1376.5)	26.2 (1236.2)
4	45.8	46.1 (1036.9)	23.1 (910.5)

の鉄道会社も数分単位での計画作成を求めており，実用的な計算時間といえる．また，マルチスレッドの計算時間はの 1 コロニーの計算時間とほぼ変わりなく理想的な並列化を達成できた．実際に反復回数を 1500 回個体数 30 で 10 回試行し性能を確認した．1 コロニーと 4 コロニーの計算で各試行で得られた解のうち最も良い評価を受けた解の平均値を表 2 に示す．マルチコロニー型 ACO により，解の探索性能が向上した．

7. おわりに

本研究では，組合せ最適化の実問題への適用を目的として，著者らが提案している H-ACO の並列化について検討を行った．H-ACO は複数コロニーで協調探索を行うため，OpenMP を用いてコロニーレベルでの並列化を行い，実問題として旅客鉄道の車両運用計画問題への適用を行った．運用計画問題への適用においては，コロニー間で同期を取る際に待ち時間が生じる．これは局所探索の処理時間のばらつきに起因し，早く終了したコロニーにあわせて局所探索を途中で打ち切ることで待ち時間を解消し，計算時間を向上させた．

今後は，Xeon Phi を用いた H-ACO の並列化を目指す．新しい並列演算処理アーキテクチャである Xeon Phi はマルチコアの数十倍のスレッド数を同時に処理でき，GPU よりも実装が容易でもある．コロニー単位の並列化に加え，さらに個体単位の並列化を行うことで更なる計算の高速化が期待できる．H-ACO の他の実問題への適用も今後展開して行く．

謝　辞

本研究は科学研究費補助金（若手研究（B）課題番号 23710170）の助成を受けて行われた．ここに謝意を表す．

文　献

[1] M. Doringo and T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.
[2] C. Twomey, T. Stützle, M. Doringo and M. Manfrin, “An analysis of communication policies for homogeneous multi-colony ACO algorithms,” *Information Science*, vol.180, issue 12, pp.2390-2404, 2010.
[3] S. Benkner, K. F. Doerner, R. F. Hartl, G. Kiechle and M. Lucka, “Communication strategies for parallel cooperative ant colony optimization on clusters and grids,” *PARA 2004*, pp.3-12, 2004.
[4] (財) 鉄道総合技術研究所 運転システム研究室, 鉄道のスケジューリングアルゴリズム, NTS, 2005.
[5] 辻 康孝, 黒田真弘, 井本善敬, 近藤英二, “アントコロニー最適化法による旅客鉄道の車両運用計画,” 日本機械学会論文集 C 編, vol. 76, no. 762, pp. 397-406, 2010.
[6] 辻 康孝, 黒田真弘, 井本善敬, 近藤英二, “複数コロニーを用いたアントコロニー最適化法による旅客鉄道の車両運用計画,” 2010 年度日本機械学会年次大会講演論文集, vol. 4, pp. 275-276, 2010.
[7] L. M. Gamberdella, E. Taillard and G. Agazzi, “MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows,” *New Ideas in Optimization*, pp.63-76, 1999.
[8] 筒井茂義, “進化計算の高速化の一手法：マルチスレッドから超多スレッドプログラミングへ,” 2011 年度電気学会全国大会講演論文集, vol. 3, pp. 3.S11(9)-3.S11(12), 2011.
[9] M. Randall and A. Lewis, “A parallel implementation of ant colony optimization,” *Journal of Parallel and Distributed Computing*, vol.62, issue 9, pp.1421-1432, 2002.
[10] T. Stützle and H. H. Hoos, “$\mathcal{MAX}-\mathcal{MIN}$ ant system,” *Future Generation Computer Systems*, vol.16, no.8, pp.889-914, 2000.
[11] 福村直登, 中村達也, 西森進矢, 坂口隆, “車両運用計画自動作成アルゴリズムの開発,” 鉄道総研報告, vol. 22, no. 6, pp. 5-10, 2008.
[12] 趙鵬, 富井規雄, 福村直登, “確率的局所探索に基づく鉄道車両運用計画作成アルゴリズム,” 電気学会 交通・電気鉄道研究会資料, TER-01-54, pp.25-30, 2001.
[13] 大野明良, 西竜志, 乾口雅弘, 高橋理, 上田健詞, “定期検査制約を有する車両運用計画問題への列生成法の適用,” 電気学会論文誌 C, vol. 132, no. 1, pp. 151-159, 2012.
[14] 今泉淳, 坂井元徳, 森戸晋, “巡回セールスマン問題に対する解法を用いた鉄道の車両運用計画の作成,” スケジューリングシンポジウム 2009 年講演論文集, pp. 285-290, 2009.
[15] 北川幸弥, 辻 康孝, 黒田真弘, “階層型アントコロニー最適化法の並列化とその車両運用計画問題への適用,” 第 32 回計測自動制御学会九州支部学術講演会, pp. 61-62, 2013.

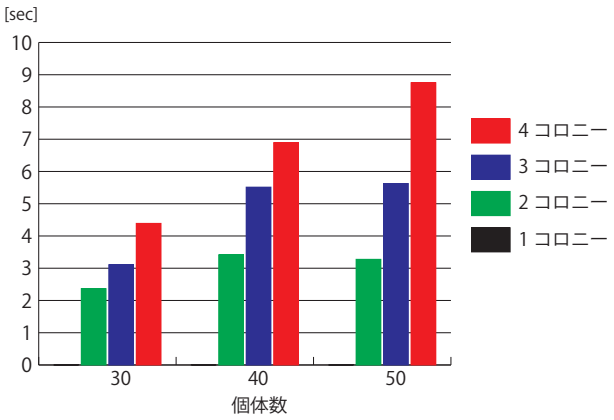


図 5　平均待ち時間（LS2 後）

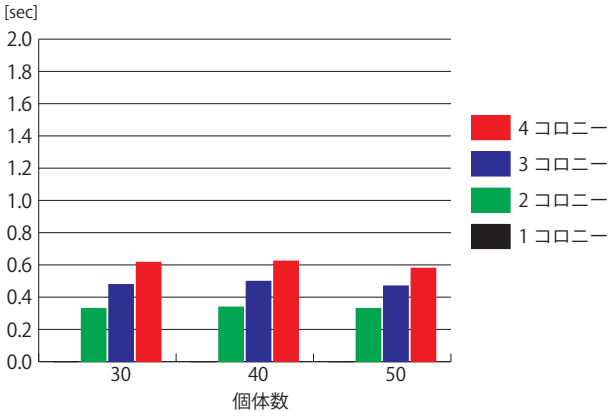


図 6　平均待ち時間（LS2 後，打ち切り適用）