Vol.2010-MPS-80 No.23
2010/9/29

# マルチコアを考慮した並列タブーサーチアルゴリズム

榎原博之[†1]　長谷川裕之介[†1]　田中裕也[†1]

近年，PC の性能向上と単価の下落が進んでいる．このことから，大規模計算を単体のスーパーコンピュータではなく，複数台の PC をネットワークに接続して構築されたクラスタ PC で計算する手法が注目されている．本研究では，同一 LAN 内に配置された複数台の PC により構築された PC クラスタ環境において，組合せ最適化問題の代表例である巡回セールスパーソン問題（TSP）を，メタヒューリスティック手法の 1 つであるタブーサーチを用いて，近似解を求める並列アルゴリズムを提案する．本研究における提案アルゴリズムの特徴は，MPI によるタブーリスト共有と，マルチコアプロセッサ上のマルチスレッドによる並列探索と，解の交叉により局所最適解からの脱出である．

# Parallelized Tabu Search Algorithms on the Multi-core Processer

Hiroyuki Ebara,[†1] Yunosuke Hasegawa[†1]
and Yuya Tanaka [†1]

In recent years, the improvement in performance and the falling in price of PCs are progressing. Therefore, not a supercomputer but a PC cluster, in which PCs are connected to the LAN, attracts attention. In this research, we propose a parallel algorithm which calculates an approximation solution for the travelling salesperson problem (TSP) using the tabu search on a PC cluster. The features of our proposed algorithm are the introduction of a shared tabu list by communication that is implemented by MPI and multi-thread on multi-core, thus, breaking away from a local solution by a crossover of solutions.

†1 関西大学大学院 理工学研究科
Graduate School for Science and Engineering, Kansai University

## 1. はじめに

近年，PC の性能の向上と価格の下落が急速に進むにつれ，比較的高性能な PC を個人で所有することが一般化しつつある．同時に，現在ではほとんどの PC はそれ単体で使用するのではなく，何らかのネットワークに接続して使用される．ネットワーク技術の発展により通信速度も飛躍的に向上しており，通信機能を持ったアプリケーションが普及してきている．

このような PC 技術，ネットワーク技術の発展と並行して，現在多様な分野からの計算要求がある．大規模計算には，データ量が莫大な問題と計算処理に長時間要する問題がある．後者の問題の代表的なものの 1 つに組合せ最適化問題がある．組合せ最適化問題とは，制約条件と目的関数が与えられたとき，制約条件を満たす組合せの中から目的関数が最適（最大あるいは最小）になる組合せ（最適解）を求める問題である．組合せ最適化問題において，求めたい結果が厳密な最適値でなければならない場合も存在するが，一般にはある程度の精度を持つ近似解でも許容され，短時間で解を得たいという場合がほとんどである．組合せ最適化問題には問題の規模に対して指数関数的に実行時間がかかることが知られている問題が多く，長時間をかけて最適な値を出すよりも短時間で充分最適解に近い近似解を得ることが重要であり，それに対する解法も数多く研究されている[1)]．数ある近似解法の中でも，特に，メタヒューリスティック手法は多くの問題に適応できる汎用的な解法であることから幅広く研究されている[2)]．本研究ではその 1 つである，多様な組合せ問題に適応可能であり，様々な実装が可能であるタブーサーチを用いる．

本研究では，組合せ最適化問題の代表例である巡回セールスパーソン問題に対してタブーサーチを用いるとこにより，良質な近似解を効率的に求めることを目的とする．本研究での提案手法での特徴は，二重の並列化である．並列化手法の一方は，MPI 通信によるマルチプロセス化であり，これによって解の多様性を実現している．もう一方は，マルチコア CPU を搭載した PC 上でのマルチスレッド化であり，このスレッド間で情報を共有している．情報の共有は共有メモリを通して行われ，MPI 通信より一層高速な通信が可能となっている．探索プロセスは，最終的に CPU プロセッサ内に搭載されているコアごとに，解空間を分割することなく振り分けられ実行される．我々はこれら提案手法を PC クラスタ上で実際に実験を行い，その有効性を検証する．

## 2. 巡回セールスパーソン問題

巡回セールスパーソン問題とは，訪問対象となる $n$ 個の都市を一度ずつ訪問して出発し

Vol.2010-MPS-80 No.23
2010/9/29

た都市に戻る巡回路の中で距離が最小のものを求める問題である[3]．

$n$ 個の都市の集合 $V = \{1, ..., n\}$ と都市 $i$ と都市 $j$ の間の距離を $C_{ij}$ とすると，目的関数 $f(x)$ を最小化する式は以下のように数式化できる：

$$f(x) = \sum_{k=1}^{n-1} C_{x(k)\ x(k+1)} + C_{x(n)\ x(1)} \quad (1)$$

ここで，$x(k) = i$ は，$k$ 番目に訪れる都市が $i$ であることを表す．

## 3. タブーサーチ

タブーサーチ（Tabu Search）はメタヒューリスティック手法の 1 つであり，以下のアルゴリズムに従って解を探索する．

(1) 初期解を生成する
(2) 近傍探索を繰り返し，タブーリストに含まれない遷移可能な解に遷移する
(3) 解の遷移と同時に，その情報をタブーリストに保存する
(4) 局所最適解に至っても，解の遷移を許し，探索を続ける
(5) 指定の終了条件になれば，探索を終了する

タブーサーチでは，解の探索過程において局所最適解に至っても終了条件が満たされるまで解を遷移し探索を続行する．探索をくり返していると遷移元の解に戻ってしまう可能性がある．タブーサーチには，これを防ぐ仕組みとしてタブーリストがある．

本研究では，近傍探索法と解の遷移方法として 2-*opt* 法を，初期解生成方法として最近近傍法を用いる．

## 4. 並列システムとその通信機能

PC クラスタ[4] とは，サーバを中心にネットワークで複数の PC が接続されているシステムである．

PC クラスタに属する PC は互いに通信を行い，クラスタ全体で 1 つの処理を行うことができるという特徴を持つ．多くの場合 PC クラスタは，LAN に代表されるような外部から遮断された独自のネットワーク環境内で構築される．このため，一般的な PC クラスタ環境において，計算処理を行う際すべての PC は計算のために占有され，高速な処理を実現する．PC クラスタ環境構築ソフトウェアの代表例として SCore[5] が挙げられる．本研究では，この SCore を用いることで MPI 環境を構築する．

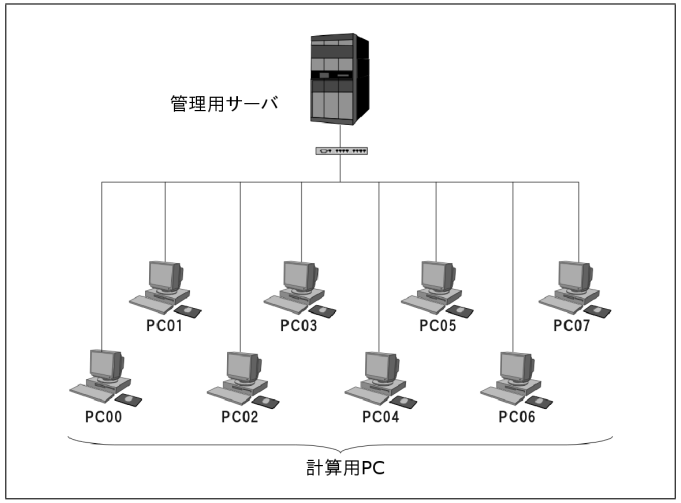


**図 1** 実験システム
Fig. 1 Experiment system

## 5. 提 案 手 法

### 5.1 実験システム

本研究で用いる実験環境は，PC クラスタコンソーシアム[5] が配布している MPI 環境構築フリーソフトウェアである SCore を用いて構築されている．これにより，研究室にある 1 台のサーバと，8 台の計算処理用 PC がギガネットイーサスイッチを介して同一 LAN 内で接続されており，互いに MPI 通信を行うことができる（図 1）．

### 5.2 提 案 手 法

本研究では，巡回セールスパーソン問題に対して，タブーサーチを並列化することにより，解空間の探索範囲を広げ，より良い近似解を得ることを目的とする．主要な並列化手法と提案手法は，MPI 通信による並列化，スレッド間の共有メモリを用いたマルチコア通信，そして解の交叉である．

MPI 通信により，複数台の PC が情報を共有し協力することで，単体の PC の場合よりも広範囲な探索を可能とする．

マルチコアによる並列化は，効率的な探索を可能とする役割を持つ．PC ごとに振り分け

られた計算ジョブは，マルチスレッドを用いることで，各 PC に搭載される CPU のコアごとに処理される．これによりスレッド数，つまり PC に搭載されるコア数に準じた効率的な探索が可能となる．

解の交叉は，解探索が局所最適解に陥った場合，そこからの脱出を可能とするために用いる．長時間解探索を行うと，探索時間とともに新たなより良い解を見つけることが困難となる．本研究における解の交叉は，こういった解探索の停滞を感知し，大幅に解に変化を与えることで，解の更新頻度の低下を防いでいる．

以下に各提案手法についての詳細な説明を示す．

(1) MPI 通信による並列化

提案手法における計算処理は，まず PC クラスタのシステムを管理するサーバが，接続されている LAN 内のすべての計算用 PC にジョブを投入することで開始される．ジョブは各計算用 PC1 台につき 1 つが割り当てられ，実行される．管理サーバは，このジョブを計算用 PC に振り分ける作業と，探索を終えた各計算用 PC から最終的な計算結果を集める作業を行う．探索が開始されると，各計算用 PC は他のすべての計算用 PC に対し，MPI による通信を行うことで情報を共有する．このように，探索はマルチプロセスで実行されるが，解空間は分割することなく，共有される情報を除いてはそれぞれ独立に探索プロセスを実行する．なお，通信は非同期に行われるため，通信処理によって探索プロセスが中断されることはない．

この PC クラスタ全体で計算用 PC が通信・共有する情報の内容は，タブーリストである（共有タブーリスト）．計算用 PC では，タブーサーチを行うために各々個別にタブーリストを保有しており（ローカルタブーリスト），共有タブーリストはこのローカルタブーリストから生成される．ローカルタブーリストにはタブーサーチでの解の遷移情報が記録されているが，探索の多様性のために，タブー期間ごとに過去のリスト情報が消去される．つまり，タブー期間をまたがって，過去にタブーとして登録された解にも，再び遷移する可能性がある．

経験的に，同一の探索アルゴリズム下において，タブー期間をまたがって重複する解の遷移の種類には偏りがあり，同じような遷移を多数繰り返す場合がある．これは単一のプロセスに留まらず，マルチプロセスで探索を行った場合も同様であり，複数の探索プロセスで重複した遷移が行われている場合がある．このような重複した探索処理を行ってしまう事態を避けるために，ローカルタブーリストに複数回登録された解の遷移情報を抽出し，共有タブーリストとして全ての計算用 PC と共有する．これにより，探索の重複を避け，探索の多様性を実現している．

(2) マルチコア・マルチスレッドによる並列化

本研究で用いる計算用 PC は，複数のコアを搭載しており，提案手法ではこのマルチコアの性能を利用する．管理サーバによって各計算用 PC に振り分けられた計算ジョブは，各々の計算用 PC で個別に探索を開始する．この中で，探索ジョブは計算用 PC に搭載されているコア数と等しい数のスレッドを生成し，それぞれのスレッドで探索処理全体を並列に実行する．このスレッドの生成と探索処理の割り振りをプログラムにより明示的に行うことで，各コアの使用率を平滑化させており，計算資源を有効に利用する．

また，マルチコアはメモリを共有しているため，コアごとに割り振られたスレッド間ではメモリを完全に共有しており，データを高速に共有することができる．このメモリ共有の利点を活かし，同一計算用 PC 内においては，各スレッドで実行される計算プロセス間で同じローカルタブーリストを使用している．ローカルタブーリストを複数スレッドで共同利用することにより，単一コアと比較して効率的な解の探索を可能にしている．

上記で述べた共有タブーリストとの違いとして，タブーとして登録される頻度が挙げられる．共有タブーリストでは重複した遷移情報のみを抽出して利用され，情報量が一定数に達した段階で他プロセスに情報が送信されるのに対し，ローカルタブーリストは解の遷移が行われ，タブーとして登録された際，同一計算用 PC に含まれる他プロセスに反映される．

(3) 解の交叉

巡回セールスパーソン問題では，すべての都市を一度ずつ通るという条件があるため，Grefenstette らが遺伝的アルゴリズムのために考案した順序表現による交叉を用いる[6]．まず，次に巡回する都市が，残っている都市リストの中で何番目に位置するかを順番に列挙していく．例えば，パス表現で表した親 1{a,e,d,c,b} は，順序表現で表すと {1,4,3,2,1}，親 2{d,b,a,e,c} は，{4,2,1,2,1} となる．順序表現よって得られた 2 つの親を 2 番目と 3 番目の間で 1 点交叉すると，{1,4,｜1,2,1},{4,2,｜3,2,1} が得られ，パス表現に戻すと，子 1{a,e,b,d,c}, 子 2{d,b,e,c,a} となり，条件を満たした 2 つの新たな解ができる（図 2）．この交叉を用いた場合，子 1 は交叉地点までの親 1 の情報，子 2 は交叉地点までの親 2 の情報のみを受け継いでおり，交叉した地点以降の都市は不規則に並び変わっている．本提案手法において，解が停滞した場合，

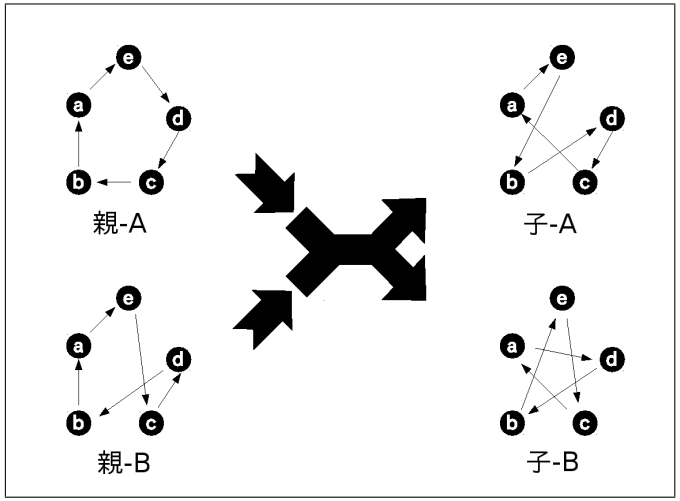


**図 2** 解の交叉
Fig. 2 Crossover

順序表現を用いた解の交叉により，解の経路が大幅に修正される．

### 5.3 提案アルゴリズム

以下に提案アルゴリズムを詳述する．

(1) 初期解の生成
初期解の生成は，最近近傍法を用いて生成される．

(2) 探索
初期解の生成後，解探索が開始される．このフェーズは終了条件で探索が終了するまで繰り返し実施される．解探索は，探索状況や探索戦略によって「局所探索」か「タブーサーチ」どちらの手法で探索を行うか変化する．

(a) 局所探索
2-*opt* 法による局所探索が行われ，必ず改善方向にのみ進む．

(b) タブーサーチ
局所探索と異なり，改悪方向にも探索を行う．また改悪許容率を満たしながらも，選択された枝がタブーであるかどうかを判断し，タブーとして登録されていた場合枝交換は行われない．提案手法において，このとき参照されるのはローカルタブーリストと共有タブーリストの両方である．

(3) 解の停滞を感知
局所最適解が一定回数探索を行っても更新されていないことを感知すると，長期的な解の停滞であると見なされ，これを打開するために交叉のフェーズへと移行する．

(4) 解の交叉
順序表現を用いた解の交叉を行う[6]．

(5) 最終終了条件
終了条件は探索時間によって決定し，あらかじめ設定しておいた時間になると探索を終了し，最も良い解を出力する．

## 6. 実験結果

### 6.1 実験環境

本研究で使用した実験システムの詳細を示す．サーバ 1 台と CPU にコアを 2 つ持つ計算用 PC8 台で実験を行う．各々の性能を表 1，2 に示す．

### 6.2 実験結果

本研究では，巡回セールスパーソン問題のベンチマーク問題例を配布する TSPLIB[7] より，575 都市，783 都市，1379 都市の 3 種類の問題を取り上げて実験を行う．TSPLIB で配布されている問題例はすべて最適値が既に算出されている．

探索時間はすべて 3,600 秒であり，実験における最適解と平均値は合計 6 回の探索結果か

**表 1** サーバの性能
Table 1 Performance of Server

| | |
|---|---|
| CPU | Intel Xeon X5470 3.33GHzX4 |
| Memory | 4GB |
| OS | CentOS 5.4 |
| MPI | SCore version 7 Beta 5 |

**表 2** 計算用 PC の性能
Table 2 Performance of Calculating PC

| | |
|---|---|
| CPU | Intel Core2Duo E6850 3.00GHzX2 |
| Memory | 4GB |
| OS | CentOS 5.4 |
| MPI | SCore version 7 Beta 5 |

ら算出する．解の停滞の判断方法は，過去に 100 回連続して局所最適解が更新されなかった場合，解の停滞であると判断される．これらのパラメータは予備実験により決定した．それぞれの問題例に対する実験結果を表 3 に示す．表の値は各実験において算出した解の総距離と既知最適値との誤差を百分率で示したものである．

並列化の手法別に 3 種類のケースで実験を行っている．上から順に「Stand Alone」は各計算処理用 PC が MPI，マルチコアどちらの並列化も行わない場合を示す．「use MPI and don't use multi-core」は MPI による並列化のみを行った場合を示す．「use MPI and use multi-core」は MPI による並列化とマルチコアによる並列化を行った提案手法を示す．

図 3，4，5 は表 3 について，各問題例ごとに表をグラフ化したものである．縦軸は解の値を示し，横軸は各並列化手法を示す．点線が平均値を示し，実践が最小値を示す．

表 3 と図 3，4，5 が示す実験結果より，MPI 通信を用いた並列化手法と，マルチコア上でのマルチスレッド並列化手法を組み合わせた提案手法が，より良質な解の導出に有用であることが示された．

### 6.3 考　　察

並列化を行わない Stand Alone 方式と MPI 並列化のみを行う方式との結果の改善率と，同様に Stand Alone 方式と提案手法との結果の改善率の平均値・最小値の場合で比較検討する．

以下の図 7，6 にそれらを示す．図 7，6 において，「Stand Alone / Use MPI」は並列化を行わない Stand Alone 方式と MPI 並列化のみを行う方式との結果の改善率を示し，「Stand Alone / Use MPI & Multi-Core」は Stand Alone 方式と提案手法との結果の改善率を表す．

平均値に関する結果を示す図 6 より Stand Alone / Use MPI の項目に関しては rat783 の改善率が最も良いことがわかる．しかし Stand Alone / Use MPI & Multi-core の項目に関しては nrw1379 の改善率が最も良くなっている．

**表 3**　rat575，rat783，nrw1379 の実験結果
Table 3　Result of rat575, rat783 and nrw1379

| | | rat575 | rat783 | nrw1379 |
|---|---|---|---|---|
| Stand Alone | ave | 1.87 | 2.59 | 3.05 |
| | min | 1.16 | 1.91 | 2.68 |
| use MPI and don't use multi-core | ave | 1.89 | 2.46 | 2.95 |
| | min | 1.18 | 1.75 | 2.59 |
| use MPI and multi-core | ave | 1.74 | 2.38 | 2.78 |
| | min | 0.99 | 1.71 | 2.07 |

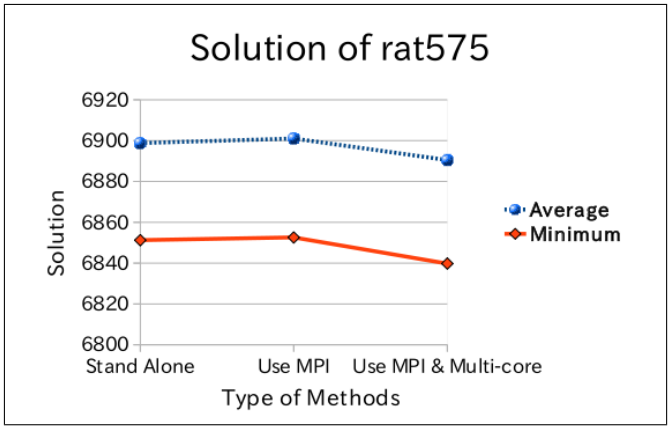


**図 3**　rat575 の結果
Fig. 3　Solution of rat575

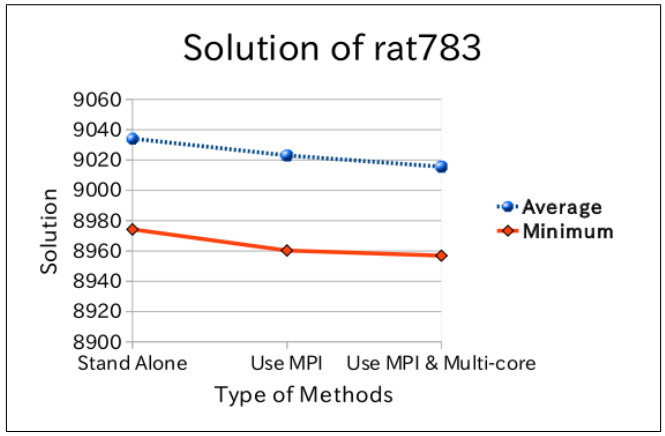


**図 4**　rat783 の結果
Fig. 4　Solution of rat783

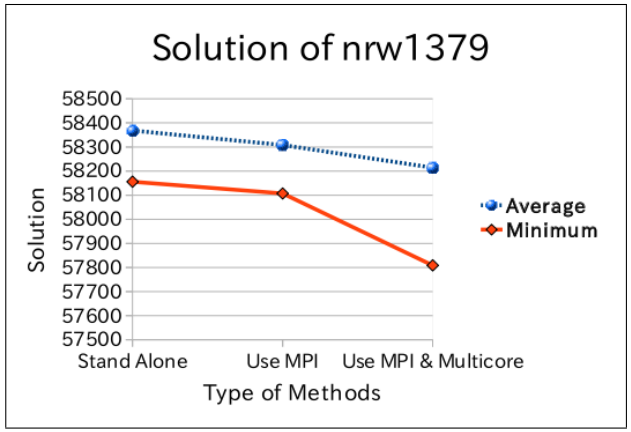


**図 5** nrw1379 の結果
Fig. 5 Solution of nrw1379

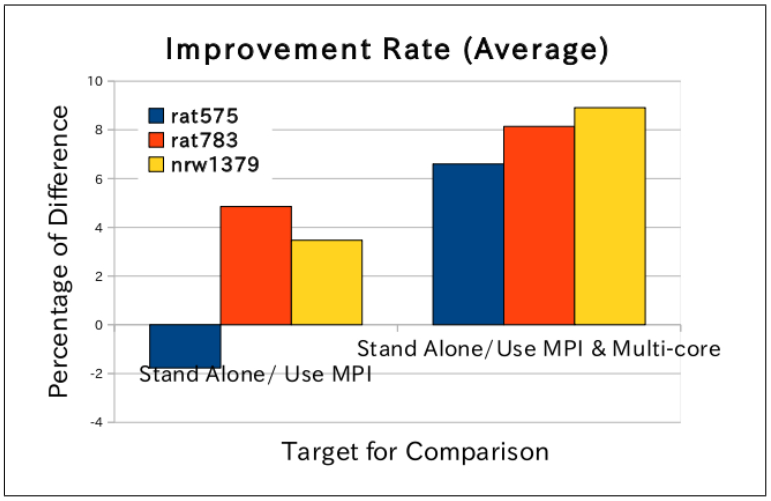


**図 6** 改善率（平均値）
Fig. 6 Improvement Rate (Average)

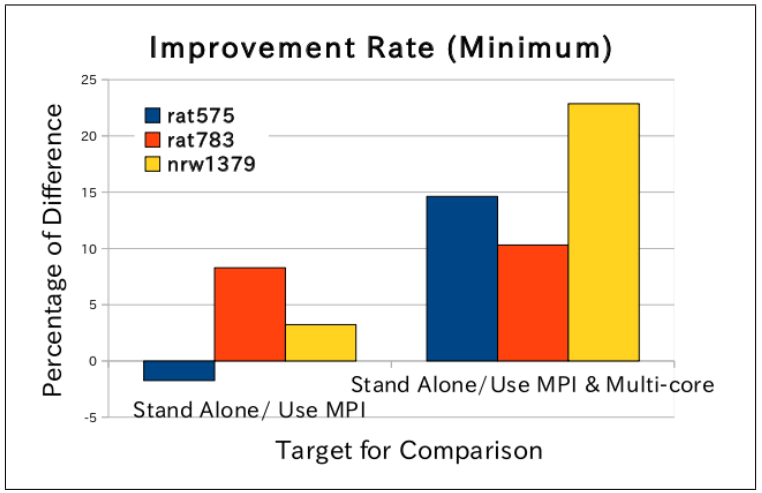


**図 7** 改善率（最小値）
Fig. 7 Improvement Rate (Minimum)

最小値に関する結果を示す図 7 では，さらに改善率が大きくなっている．

これらの結果から，提案手法は，改善率において平均でも 6-9%，最小値では 10-23%良くなっていることが分かる．

また，MPI による並列化だけではなく，マルチコアによる並列化も併用した手法が確実に良質な結果を算出することが明らかとなった．

MPI による並列化により多様性を，マルチコアによる並列化により効率性を増大させるという，各々の役割を通信様式によって分担する本実験の提案手法が，より良質な解の導出に有用であることが確認出来た．

## 7. おわりに

本研究では，MPI 通信とマルチコアに振り分けたマルチスレッドを用いた並列環境において，巡回セールスパーソン問題に対する並列タブーサーチアルゴリズムを提案した．本提案手法では，解の多様性を高め解の停滞を防ぐために，共有タブーリストを導入し，また同一計算処理用 PC 内でメモリを共有させたマルチプロセスによる探索を実現し，さらに遺伝的アルゴリズムに倣った解の交叉を行うことで，良質な解が得られることを計算機実験に

Vol.2010-MPS-80 No.23
2010/9/29

より検証した．実験結果から，並列化を行わない場合に比べ，良い近似解が得られており，多様な解への探索と重複の少ない探索が実現できていることが分かった．

## 参 考 文 献

1) S.Sait and H.Youssef: 組合せ最適化アルゴリズムの最新手法，丸善 (2002).
2) C.R.Reeves: モダンヒューリスティックス 組合せ最適化の先端手法，日刊工業新聞社 (1997).
3) 山本芳嗣，久保幹雄：巡回セールスマン問題への招待，朝倉書店 (1997).
4) T.L.Sterling, J.Salmon, D.Becker and D.F.Savarese: PC クラスタ構築法 Linux によるベオウルフシステム，産業図書株式会社 (2001).
5) PCCC: PC Cluster Consortium. http://www.pccluster.org/.
6) P.Larraaga, C.M.H.Kuijpers, R.H.Murga, I.Inza and S.Dizdarevick: Genetic algorithms for the travelingsalesmanproblem, *the First International Conference on Genetic Algorithms*.
7) TSPLIB: . http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/.