カニングアントを用いたACOの2次割当て問題への適用とその並列化

筒井茂義[†1] 劉 力綺[†1] 小島基伸[†2]

筆者らは先にカニングアントシステムと呼ぶ新しい Ant Colony Optimization アルゴリズムを提案し，巡回セールスマン問題を用いて評価を行いその有効性を確認した．本論文は，カニングアントシステムの 2 次割当て問題への適用に関するものである．2 次割当て問題は巡回セールスマン問題と同様 NP 困難な問題であるが，順序表現問題の中でも最も困難な問題の 1 つと考えられている．本論文では，カニングアントシステムの 2 次割当て問題への適用方法について述べ，Ant Colony Optimization アルゴリズムの中で最も有効な手法の 1 つとされている Max-Min Ant System との比較を行い，カニングアントシステムが 2 次割当て問題においても有効な Ant Colony Optimization アルゴリズムの 1 つであることを示す．また，カニングアントシステムの並列化の一方法についても述べる．

Cunning Ant System for Quadratic Assignment Problem with Local Search and Parallelization

Shigeyoshi Tsutsui,[†1] Lichi Liu[†1] and Motonobu Kojima[†2]

The previously proposed cunning Ant System, a variant of the Ant Colony Optimization algorithm, worked well on the Traveling Salesman Problem and the results showed that the cunning Ant System is a promising Ant Colony Optimization algorithm on Traveling Salesman Problems. In this paper, we apply the cunning Ant System to solving the Quadratic Assignment Problem and compare it with Max-Min Ant System. We also discuss a parallelization method of the cunning ant system. The experimental results showed the cunning Ant System worked very well on the Quadratic Assignment Problem and it may be one of the most promising Ant Colony Optimization algorithms on the Quadratic Assignment Problem as well.

1. はじめに

アリの行動にヒントを得た Ant Colony Optimization（ACO）アルゴリズムは，各種の組合せ最適化問題に応用され，大きな成果が得られている．ACO アルゴリズムの基礎モデルは，Dorigo らが巡回セールスマン問題（Traveling Salesman Problem，以下 TSP）の解法に適用した Ant System（AS）[5] に集約されている．一般にメタヒューリスティックスを用いた探索では，探索過程における多様化（exploration）と集中化（exploitation）とのバランスをとることが重要であり，ACO アルゴリズムでも同様である．このような視点からの ACO アルゴリズムの拡張の研究も活発に行われ，ランクベース Ant System（AS_{rank}）[2]，Ant Colony System（ACS）[4]，Max-Min Ant System（MMAS）[16] や経路選択にランダムネスを導入するアルゴリズム[21] などの研究がある．この中で，MMAS が最も有力な ACO アルゴリズムの 1 つであることが報告されている[6]．

筆者らは，先に，解候補を生成する際，過去に得られた解の一部分を借用するという行動をとるエージェントを導入した ACO アルゴリズムとして，カニングアントシステム（cunning Ant System，以下 cAS）の提案を行った[19),22]．そこでは，TSP をテスト問題に用いて評価を行い，cAS が高性能な ACO アルゴリズムの 1 つであることを示した．

本論文では，cAS を 2 次割当て問題（Quadratic Assignment Problem，以下 QAP）に適用する方法について述べ，主に MMAS との比較を通して cAS

†1 阪南大学大学院企業情報研究科
Graduate School of Corporate Information, Hannan University
†2 シグネット
Cygnet

が QAP においても有効な ACO アルゴリズムの 1 つであることを示す．QAP は流通コストを考慮した工場などの配置問題やビル内における部門の配置問題など，多くの応用を有する最適配置問題を一般化したモデルである．しかし，QAP は，TSP のようにノード間の関係を示すエッジに関する情報を有効に使うことができないため，TSP に比べて困難な問題であるといわれている[15),16)]．なお，ACO アルゴリズムの QAP への適用の研究には，AS-QAP[12)]，MMAS-QAP[16)]，ANTS-QAP[11)] などが代表的であり，遺伝的アルゴリズム（GA）などの他のメタヒューリスティックスよりも良い結果が報告されている[16)]．

以下，本論文ではまず 2 章において cAS の概要を振り返る．次に 3 章において QAP への cAS の適用法を述べる．4 章ではテスト問題を用いて行った実験結果について述べ，MMAS と比較する．最後に 5 章では，cAS の高速化を図るための並列化の一方式を提案し，その効果を述べる．

2. カニングアントシステム（cAS）の概要[19),22)]

2.1 カニングアント

通常の ACO アルゴリズムでは，各エージェントにより生成される解はフェロモン濃度 $\tau_{ij}(t)$ に基づいて確率的に生成される．ただし，$\tau_{ij}(t)$ は解の順序表現におけるノード i，j 間のフェロモン濃度である．TSP の例では，現在ノード（都市）i にいるエージェントが次に訪問するノード j を選ぶ確率が次式で与えられる[5)]．

$$p_{ij}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{s\in N(i)}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}]^{\beta}} & \text{if } j \in N(i), \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

ただし，η_{ij} はノード i，j 間のヒューリスティック値，$N(i)$ は，ノード i にいるエージェントがまだ選択できるノードの集合であり，α と β はフェロモン濃度 $\tau_{ij}(t)$ とヒューリスティック値 η_{ij} との重要性の度合いを調整するパラメータである．TSP の場合は一般に $\eta_{ij} = 1/d_{ij}$ が使われ（ただし，d_{ij} は，ノード i，j 間の距離），また適切なヒューリスティック値が存在しない場合は $\eta_{ij} = 1$ とされる．なお，ACS[4)] では，式 (1) の利用を確率的に適用する疑似サンプリング法が提案されている．

cAS では，カニングアント（cunning ant，以下 *c-ant*）と呼ぶエージェントを導入している．*c-ant* は，従来のエージェントと異なり，新しい解候補を生成の際に，過去の探索において存在する解の一部分（部分解）を借用し，解の残りの部分の生成には従来と同様，式 (1) に基づいてフェロモン濃度を利用する．このようなエージェントを抜け目のないアリという意味でカニングアントと呼んだ．狡をするエージェントという意味も言外に込められている．カニングアントに借用を許すエージェントをドナーアント（donor ant，以下 *d-ant*）と呼ぶ．

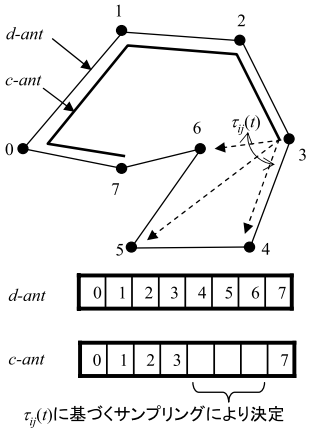

図 1 TSP における *c-ant* と *d-ant*
Fig. 1 *c-ant* and *d-ant* in TSP.

図 1 は，TSP における *c-ant* と *d-ant* との関係の例示である．なお，ここでは *c-ant*，*d-ant* と呼ぶ場合，エージェント自身と同時にその解のことも指している．この例では，*c-ant* は経路 7→0→1→2→3 を *d-ant* から借用し，残りのノード 4，5 および 6 の経路は，式 (1) に基づき確率的に生成する．

このように *c-ant* は，経路を一部借用するという方法により，探索過程においてポジティブフィードバックによる集中化を緩和し，フェロモン濃度 $\tau_{ij}(t)$ の初期収束を抑止する．

2.2 コロニーモデル

cAS では，**図 2** に示すような m 個のユニットからなるコロニーモデルを用いる．各ユニット k（$k = 1, 2, \ldots, m$）は 1 つのエージェント $ant^*_{k,t}$ から構成され，その経路が保存されている．フェロモン濃度はコロニー全体の記憶であり，m 個のユニットで共有される．経路形成は各ユニットのエージェント $ant^*_{k,t}$ が $d\text{-}ant_{k,t}$ となり，$c\text{-}ant_{k,t+1}$ が生成される．次に，$d\text{-}ant_{k,t}$ と $c\text{-}ant_{k,t+1}$ とが比較され，良い方の経路を持つものがそのユニットの次の繰返し $t+1$ におけるエージェント $ant^*_{k,t+1}$ となる．このように cAS のコロニーモデルでは，各ユニットの最良解が $ant^*_{k,t}$ と

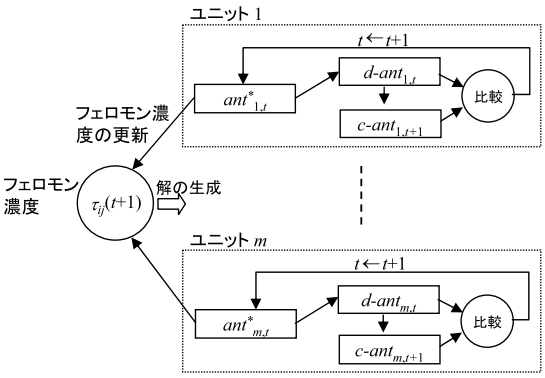

図 2 *c*AS のコロニーモデル
Fig. 2 Colony model for *c*AS.

して保存される．

2.3 フェロモン濃度の更新

フェロモン濃度の更新は，$ant^*_{k,t}$（$k = 1, 2, \ldots, m$）によって行われ，AS[5] と同様 $\tau_{ij}(t+1)$ は次式によって更新される．

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta^* \tau^k_{ij}(t) \qquad (2)$$

$$\Delta^* \tau^k_{ij}(t) = \begin{cases} 1/C^*_{k,t} & : \text{if } (i,j) \in ant^*_{k,t}, \\ 0 & : \text{otherwise} \end{cases} \qquad (3)$$

ただし，ρ（$0 \leq \rho < 1$）は，蒸発係数，$\Delta^* \tau^k_{ij}(t)$ は $ant^*_{k,t}$ により i，j 間の経路に放出されるフェロモン量，$C^*_{k,t}$ は $ant^*_{k,t}$ の経路長である．*c*AS では，フェロモン濃度の更新は式 (2) に基づき m 個の $ant^*_{k,t}$ によって行われるが，ここで，MMAS と同様 $[\tau_{min}, \tau_{max}]$ の区間を定義する．*c*AS では，τ_{max} と τ_{min} は次式で定義される．

$$\tau_{\max}(t) = \frac{1}{1-\rho} \times \sum_{k=1}^{m} \frac{1}{C^*_{k,t}} \qquad (4)$$

$$\tau_{\min}(t) = \frac{\tau_{\max} \cdot (1 - \sqrt[n]{p_{best}})}{(n/2 - 1) \cdot \sqrt[n]{p_{best}}} \qquad (5)$$

式 (5) は，MMAS と同じであり，p_{best} は τ_{min} を決定するのに導入された制御パラメータである．*c*AS のフェロモン放出は m 個の $ant^*_{k,t}$（$k = 1, 2, \ldots, m$）により行われるのに対して，MMAS では，最良解エージェント（*best-so-far* エージェントまたは *iteration-best* エージェント）によってのみフェロモンが放出される．したがって，式 (4) は MMAS の τ_{max} の定義から変形されたものとなっている（詳しくは文献 16) を参照）．

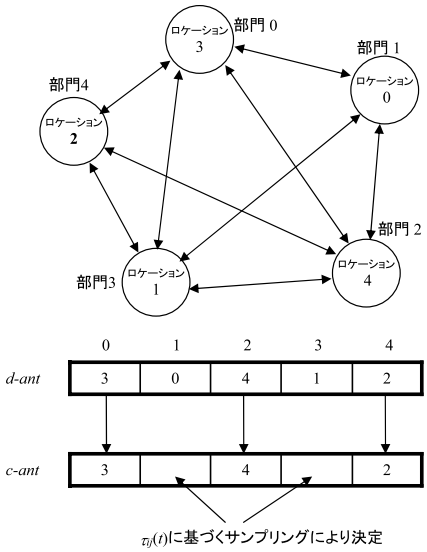

図 3 QAP における *c-ant* と *d-ant*
Fig. 3 *c-ant* and *d-ant* in QAP.

3. 2 次割当て問題への *c*AS の適用

3.1 2 次割当て問題

2 次割当て問題（QAP）は，n 個からなる部門に n 個のロケーションを次式で定義される値が最小になるような順列 ϕ を求める問題である．

$$f(\phi) = \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} b_{rs} a_{\phi(r)\phi(s)} \qquad (6)$$

ただし，$A = (a_{pq})$ および $B = (b_{rs})$ は，それぞれ，$n \times n$ のマトリックスであり，ϕ は $\{0, 1, \ldots, n-1\}$ の順列である．マトリックス A はロケーション p とロケーション q 間の距離を，マトリックス B は部門 r と部門 s 間の流量を，それぞれ表している．このように QAP は部門間の流量と距離との積の和が最小になるように各部門を各場所に割り当てるという問題であり，最適配置問題を一般化したものである．

3.2 2 次割当て問題における *c-ant* と *d-ant*

QAP における *c-ant* は，TSP における *c-ant* と $\tau_{ij}(t)$ の定義において以下のように異なっている．TSP では，フェロモン濃度 $\tau_{ij}(t)$ は各都市間のエッジ (i, j) に割り当てられ，エッジ (i, j) の好ましさの度合いと定義されている．これに対して QAP では，$\tau_{ij}(t)$ は部門 i にロケーション j を割り当てる好ましさの度合いと定義される[16]．

本論文でも $\tau_{ij}(t)$ にこの定義を用いる．**図 3** は QAP における *c-ant* と *d-ant* との関係の例示である．この例では，*c-ant* は部門 0，2 および 4 のノード値（ロ

ケーション）を *d-ant* から借用している．残りの部門 1 および 3 のノード値（ロケーション）は次式に従って確率的に決定される．

$$p_{ij}(t) = \frac{\tau_{ij}(t)}{\sum_{k \in N(i)} \tau_{ik}(t)} \tag{7}$$

本論文の QAP では，ヒューリスティックは適用していないので，式 (7) は式 (1) において $\eta_{ij} = 1$ としたものである．このように *c-ant* の利用は，解の一部分のみが新たに生成されるので，TSP において観察されたと同様[19),22)] 探索過程においてポジティブフィードバックによる集中化を緩和し，フェロモン濃度 $\tau_{ij}(t)$ の初期収束を抑止することが期待できる．コロニーモデルは，TSP で用いた図 2 と同様である．

3.3 サンプリング法

c-ant が $\tau_{ij}(t)$ に基づいてサンプルするノードの数を l_s，*d-ant* から借用するノードの数を l_c とすると，$l_c = n - l_s$ である．ここで，TSP における *c*AS で導入したと同様，l_s の平均値 $E(l_s)$ が $E(l_s) = n \times \gamma$ となるような制御パラメータ γ を導入し，そのような条件を満たす l_s の分布を決めるものとして，TSP で用いたのと同じである以下の確率密度関数 $f_s(l)$ を用いる[19),22)]．

$$f_s(l) = \begin{cases} \frac{1-\gamma}{n\gamma}\left(1 - \frac{l}{n}\right)^{\frac{1-2\gamma}{\gamma}} & \text{for } 0 < \gamma \leq 0.5, \\ \frac{\gamma}{n(1-\gamma)}\left(\frac{l}{n}\right)^{\frac{2\gamma-1}{1-\gamma}} & \text{for } 0.5 < \gamma < 1 \end{cases} \tag{8}$$

さて，QAP における *c-ant* の *d-ant* からの部分解の借用方法は TSP の場合と以下のように異なる．TSP では，その性質上 *c-ant* が *d-ant* から借用するノードは順序表現における連続した位置，すなわちサブツアーとなる（図 1 参照）．一方 QAP では，$\tau_{ij}(t)$ は部門 i にロケーション j を割り当てる好ましさの度合いと定義しているので，*c-ant* が借用するノードが連続した位置でなければならないという制約は存在しない．したがって，QAP では *c-ant* が *d-ant* から借用する l_c 個のノードの解表現における位置はランダムに選ぶ．また，式 (7) におけるサンプリングにおけるノードの位置の順序もランダムとする．

以上をまとめるとサンプリングは以下のとおりである．まず式 (8) に基づいて *c-ant* がサンプリングするノードの数 l_s を決定する．このとき，l_s の平均値 $E(l_s)$ は $n \times \gamma$ であり，*c-ant* が *d-ant* より借用するノードの数 l_c は $n - l_s$ となる．借用する l_c 個のノードの位置をランダムに決定し，その位置のノード値を *d-ant* から *c-ant* にコピーする．残りの l_s 個のノード値は，ランダムな位置の順序で式 (7) に基づいてサンプリングする．

4. *c*AS の 2 次割当て問題における性能評価

本章では，QAPLIB[13)] におけるテスト問題を用いて *c*AS の QAP における性能を評価する．QAP のテスト問題は，i) ランダム生成問題（randomly generated instances），ii) グリッドベース問題（grid-based distance matrices），iii) 実問題（real-life instances），および iv) 実問題風テスト問題（real-life-like instances）の 4 つに分類できる[7),16)]．本章では，MMAS の研究で用いられているこれらの分類の問題を用いて実験を行う．

実験に用いた計算機は，Opteron 280（dual core, 2.4 GHz，Socket940）プロセッサからなる Windows XP マシン（主メモリ：2 GB）である．また，*c*AS は，Java 言語で記述しており，実行環境には Java2（j2sdk1.4.2_13）を用いる．

4.1 実験方法と結果

ここでは，問題サイズ n が 35 以下の問題として，分類 i) に属するテスト問題 tai20a（$n = 20$），tai25a（$n = 25$），tai30a（$n = 30$），tai35a（$n = 35$），分類 ii) に属するテスト問題 nug30（$n = 30$），分類 iii) に属するテスト問題 kra30a（$n = 30$），kra30b（$n = 30$）および分類 iv) に属するテスト問題 tai20b（$n = 20$），tai25b（$n = 25$），tai30b（$n = 30$），tai35b（$n = 35$）の 11 個の問題を用いて MMAS と比較する．なお，問題サイズ n が 35 およびそれを超える問題を用いる実験は，4.3 節のローカルサーチを適用する場合に行う．

実験条件は *c*AS，MMAS とも以下のとおりである．解の評価回数の上限 $E_{max} = n \times 800{,}000$，エージェント数 $m = n \times 4$，蒸発係数 $\rho = 0.9$，式 (5) における制御パラメータ $p_{best} = 0.005$ とし，25 回の実験を行う．

MMAS では，フェロモン放出エージェントとして *best-so-far* エージェントまたは *iteration-best* エージェントが用いられ，それらの適用の方法でチューニングが行われる．具体的には，あるインターバル *interval* を決め，アルゴリズムの $(interval - 1)$ 回の繰返しは *iteration-best* エージェントにフェロモンを放出させ，次に *best-so-far* エージェントに 1 度フェロモンを放出させる．そしてこれを繰り返す．このようにして適切な *interval* を決めることでチューニングが行われる．本研究では *interval* 値として，2, 3, . . . , 10 を用いる 9 個の実験を行って MMAS のチューニングを行い，*Error*（(最良解 − 最適解)/(最適解)）

表 1 *c*AS と MMAS の QAP における性能．値（*Error* 値（%））は 25 回の実験の平均である
Table 1 Performance of *c*AS and MMAS on QAP. The results are average of *Error* (%) over 25 independent runs.

分類	QAP	値の項目	*c*AS		MMAS	
			γ=0.3	*non-c*AS(γ=1)	MMAS+pts	MMAS
i	tai20a	*Error (%)*	**1.006**	2.779	2.9957	3.1403
		Std	0.0421	0.8412	0.7816	1.0006
		t 値	-	15.900	18.494	15.175
		P 値	-	3.03E-14	1.05E-15	8.43E-14
	tai25a	*Error (%)*	**1.566**	3.206	3.217	3.380
		Std	**0.529**	0.007	0.638	0.714
		t 値	-	14.507	9.965	10.210
		P 値	-	2.24E-13	4.55E-13	3.50E-13
	tai30a	*Error (%)*	**1.843**	2.692	3.004	2.758
		Std	**0.316**	0.747	0.795	0.703
		t 値	-	5.230	6.786	5.935
		P 値	-	1.02E-05	1.34E-07	1.17E-06
	tai35a	*Error (%)*	**2.194**	3.015	3.690	3.600
		Std	**0.264**	0.692	0.768	0.719
		t 値	-	5.545	9.210	9.185
		P 値	-	4.50E-06	3.00E-10	3.19E-10
ii	nug30	*Error (%)*	**0.455**	1.551	1.675	1.962
		Std	**0.249**	0.874	0.998	0.909
		t 値	-	6.031	5.929	7.998
		P 値	-	1.68E-06	2.55E-06	1.04E-08
iii	kra30a	*Error (%)*	**1.147**	3.381	3.963	4.014
		Std	**0.860**	1.830	1.770	1.740
		t 値	-	5.524	7.153	7.386
		P 値	-	3.58E-06	2.42E-08	1.22E-08
	kra30b	*Error (%)*	**0.447**	2.172	2.736	2.836
		Std	**0.320**	1.528	0.836	1.359
		t 値	-	5.517	5.517	8.538
		P 値	-	8.63E-06	8.63E-06	3.76E-09
iv	tai20b	*Error (%)*	**0.000**	0.745	0.348	0.388
		Std	**0.000**	0.486	0.354	0.427
		t 値	-	7.658	4.920	4.547
		P 値	-	6.77E-08	5.09E-05	1.31E-04
	tai25b	*Error (%)*	**0.003**	0.614	1.753	2.385
		Std	**0.014**	1.301	2.194	3.232
		t 値	-	2.350	3.988	3.686
		P 値	-	2.73E-02	5.44E-04	1.16E-03
	tai30b	*Error (%)*	**0.066**	5.465	2.274	2.279
		Std	**0.074**	3.263	1.510	1.629
		t 値	-	8.272	0.304	6.787
		P 値	-	1.74E-08	1.52E-07	5.09E-07
	tai35b	*Error (%)*	**0.252**	1.948	2.453	2.472
		Std	**0.136**	1.657	1.877	1.907
		t 値	-	5.099	5.845	5.805
		P 値	-	3.24E-05	4.99E-06	5.52E-06

Std: 標準偏差

が最小になる *interval* 値として 5 を得て，これを用いた．すなわち，MMAS のフェロモン放出エージェントの選び方として，アルゴリズムの繰返しにおいて *iteration-best* エージェント 4 回に対して *best-so-far* エージェント 1 回用いた．*c*AS の γ 値は 0.3 とした．

表 1 に結果のまとめを示す．同表において，*non-c*AS は，カニング行動をとらない，すなわち $\gamma = 1$ の場合を示す．ただし，コロニーモデルは，2.2 節の図 2 のものを用いる．太字の数値は，最も良い結果を示したものである．これから *c*AS（$\gamma = 0.3$）が優れた性能を持っていることが分かる．また，*non-c*AS の結果も MMAS の結果と同程度の性能を有していることが分かり，TSP の場合と同様，図 2 のコロニーモデルが有効であることを示している．なお，同表には *c*AS（$\gamma = 0.3$）と *non-c*AS（$\gamma = 1$），MMAS+pts および MMAS（pts なし）との間の統計的検定値として，*t* 値および *P* 値を示しているが，*c*AS（$\gamma = 0.3$）と他との比較における統計的有意性は明確である．

4.2 γ の影響と収束過程の分析

表 1 では，$\gamma = 0.3$ の場合のみの *c*AS の結果を示した．ここでは，分類 i)，ii)，iii) および iv) から，それぞれ次の 1 つの問題，tai30a，nug30，kra30a および tai30b を取り上げ，γ の値を 0.1 から 0.9 まで 0.1 刻みで変化させて実験を行う．このときの *Error* の変化を平均と標準偏差を用いて**図 4** に示す．ここで，各 γ 値に対して，中点は平均値の場所を，その上下

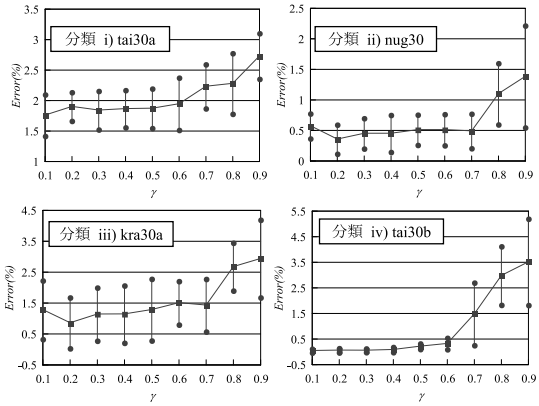

図 4 γ による $Error$ の変化
Fig. 4 $Error$ for various values of γ.

$$\tau_{ij}(t)=\begin{array}{c} \\ 0\\1\\2\\3\\4\end{array}\begin{array}{c}\begin{array}{ccccc}0&1&2&3&4\end{array}\\ \begin{pmatrix}\tau_{\min}&\tau_{\min}&\tau_{\min}&\boldsymbol{\tau_{\max}}&\tau_{\min}\\ \boldsymbol{\tau_{\max}}&\tau_{\min}&\tau_{\min}&\tau_{\min}&\tau_{\min}\\ \tau_{\min}&\tau_{\min}&\tau_{\min}&\tau_{\min}&\boldsymbol{\tau_{\max}}\\ \tau_{\min}&\boldsymbol{\tau_{\max}}&\tau_{\min}&\tau_{\min}&\tau_{\min}\\ \tau_{\min}&\tau_{\min}&\boldsymbol{\tau_{\max}}&\tau_{\min}&\tau_{\min}\end{pmatrix}\end{array}$$

図 5 QAP における $\tau_{ij}(t)$ の収束
Fig. 5 Convergence of $\tau_{ij}(t)$ in QAP.

の点は平均値から標準偏差の値だけずれた場所を示している．

この図から，*c-ant* を用いる効果が分かる．すなわち，γ が小さい値（$[0.1, 0.5]$）において $Error$ の値は良い値を持つ．γ の影響をより詳しく見るために以下では，収束過程を多様性の尺度を導入して分析する．

4.2.1 フェロモン濃度のエントロピー尺度の定義

フェロモン濃度 $\tau_{ij}(t)$ の多様性を示すものとしてエントロピー尺度 $I(t)$ を，以下のように定義する．

$$I(t) = -\frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} p_{ij}(t)\log p_{ij}(t) \tag{9}$$

$$p_{ij}(t) = \frac{\tau_{ij}(t)}{\sum_{j=0}^{n-1}\tau_{ij}(t)} \tag{10}$$

$I(t)$ の上限値は $\tau_{ij}(t)$ の各要素がすべて同じ値を持つとき，すなわちアルゴリズムの初期化（$t=0$）のときであり，その値は定義から以下となる．

$$\overline{I} = \log(n) \tag{11}$$

次に $I(t)$ の下限値であるが，$\tau_{ij}(t)$ は区間 $[\tau_{min}, \tau_{max}]$ に限定されるので，$I(t)$ が最小になるのは，**図 5** に例示するように，$\tau_{ij}(t)$ の各行の 1 つのエントリが τ_{max} に，その他のすべてのエントリが τ_{min} になったときである．このときの $I(t)$ の値は定義から，

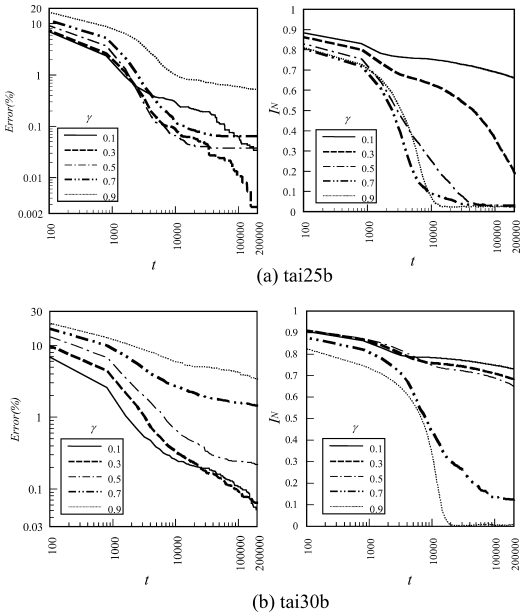

図 6 tai25b と tai30b の収束過程
Fig. 6 Convergence processes on tai25b and tai30b.

$$\underline{I} = \log(r+n-1) - \frac{r\log(r)}{r+n-1} \tag{12}$$

として得られる．ただし，$r = \tau_{max}/\tau_{min}$ である．$I(t)$ は，その定義から問題サイズ n，区間上下限値 τ_{max}，τ_{min} に依存するので，以下では，

$$I_N(t) = \frac{I(t) - \underline{I}}{\overline{I} - \underline{I}} \tag{13}$$

と規格化したエントロピー $I_N(t)$ を用いる．$I_N(t)$ は，区間 $[0.0, 1.0]$ の値をとり，n，τ_{max} および τ_{min} に直接影響されない値となる．

4.2.2 収束過程の分析

図 6 は，tai25b と tai30b の収束過程を示したものである．同図において，左側は $Error$ の変化を示し右側は規格化したエントロピー $I_N(t)$ を示している．いずれも各値は，25 回の実験の平均値である．tai25b を見ると，γ 値が 0.5，0.7 および 0.9 のときは $I_N(t)$ は繰返し数 t がそれぞれ 80,000，40,000 および 20,000 近傍でほぼ 0 になり，$\tau_{ij}(t)$ が収束していることが分かる．この繰返し数 t の値は，同図左側で $Error$ 値の減少が止まる繰返し数 t にほぼ一致している．γ 値が 0.3 のときは，$I_N(t)$ は徐々に減少し，探索が続いていることが $Error$ 値の変化からも確認できる．γ 値が 0.1 のときは，$I_N(t)$ は大きな値を維持しており，結果として $Error$ 値での収束が遅くなっている．tai30b の場合もほぼ同様な傾向を確認することができる．

以上のようにエントロピー尺度を用いた収束過程の

分析から *c-ant* エージェントを用いる効果が確認できる．すなわち，*c-ant* は，平均として $(1-\gamma)$ の割合の部分解を既存解 *d-ant* から借用し，残りの γ の割合の部分解を新たに生成する．このように γ の値により既存解の利用（exploitation）と新たな探索（exploration）のバランスを制御している．なお，極端な小さな γ 値の適用は，探索の収束を遅らせる．

4.3 ローカルサーチの適用での比較

一般にメタヒューリスティックスを用いる探索では，ローカルサーチを併用することが一般的である．そこで，本節では QAP を解く *c*AS にローカルサーチを適用した場合での MMAS との比較結果について述べる．MMAS の QAP への適用の研究[16] では，ローカルサーチとして Taillard によるロバストタブーサーチ（Robust taboo search，以下，Ro-TS）[7]（MMAS-TS）と 2OPT（MMAS-2OPT）とを適用している．QAP における 2OPT の実装方式には種々の改良方式が提案されている（たとえば，文献 20)，23)）．しかし，MMAS の文献[16] では，2OPT の実装の詳細が記述されていないので，ここでは共通に利用できる Ro-TS を *c*AS に適用する．テスト問題としては，4.1 節で用いたものよりも大きな問題を選び，結果を文献 16) で報告されているものと比較する．

4.3.1 実験方法と結果

Ro-TS をローカルサーチとして用いる *c*AS（*c*AS-TS）の実験の方法は MMAS の研究[16] で用いられたものと同じとする．すなわち，$m = 5$，$\rho = 0.8$，$p_{best} = 0.005$ とし，また，*c*AS による解生成最大個数は 250 とし，各解に短い Ro-TS を適用する．この際の Ro-TS の繰返し数は $4 \times n$ である．解生成個数を 250 としているので，アルゴリズムにおける繰返し数の最大はすべての問題において 50（$= 250/5$）である．なお，MMAS のこの設定は論理的な実行時間が，Ro-TS 単独の実行における性能比較の際の実行時間（Ro-TS 単独の場合の繰返し数は $1{,}000 \times n$ としている）とほぼ同じとなるように決められたものであり，探索時間を同一にして比較を行うために設定されたものであることに注意していただきたい．

同論文において，MMAS-2OPT の実行時間は MMAS-TS 実行時間に達したときに打ち切りとしている．また同論文では比較対象として Ro-TS 単独のほか，遺伝的アルゴリズム（GA）にタブーサーチを併用する Genetic Hybrid 法（GH）[8]，AS に 2OPT ローカルサーチを併用する Hybrid Ant System（HAS）[9] を用いているが，これらのアルゴリズムの論理的実行時間も上記の実行時間とほぼ同等となっている．

なお，Ro-TS のコードは C 言語で記述され，研究用に公開されている[18]．一方，本研究における *c*AS は Java 言語で記述している．*c*AS で生成した解にローカルサーチを適用する際，Ro-TS に効率的に引き渡すためには，両プログラムを密に結合する必要がある．Java 言語で記述されたプログラムと C 言語で記述されたプログラムとを結合する方法としては，Java Native Interface（JNI）を用いる方法があるが，本研究ではプログラムの一元管理を容易にするために，Ro-TS のコードを Java に変換して用いた．この変換では，Ro-TS のコード全体を 1 つのクラスとし，C 言語の記述された関数を Java のメッソードの形式に単純に変更することで，プログラムロジックには何らの変更も加えていない．また，プログラムを Java 言語に統一して一元管理したことによって，5 章で述べる *c*AS の並列化の実現も容易になっている．

実験に用いたテスト問題は，QAP の分類 i) に属するテスト tai35a（$n = 35$），tai40a（$n = 40$），tai50a（$n = 50$），tai60a（$n = 60$），tai80a（$n = 80$），分類 ii) に属する sko42（$n = 42$），sko49（$n = 49$），sko56（$n = 56$），sko64（$n = 64$），sko72（$n = 72$），sko81（$n = 81$），sko90（$n = 90$），分類 iii) に属する ste36a（$n = 36$），ste36b（$n = 36$）および分類 iv) に属する tai35b（$n = 35$），tai40b（$n = 40$），tai50b（$n = 50$），tai60b（$n = 60$），tai80b（$n = 80$），tai100b（$n = 100$）の 20 問題である．

実験回数は，MMAS の研究[16] では 10 回としているので，ここではその回数にあわせて 10 回とし，4.1 節の実験と同様 *Error* で評価した．

表 2 に結果をまとめる．*c*AS-TS における γ 値は 0.4 および 0.8 とした．同表には，MMAS-TS，MMAS-2OPT，GH および Ro-TS 単独の結果も示しているが，これらの数値はいずれも文献 16) に記述されているものである．まず，同表において *c*AS-TS と MMAS-TS との比較を見る．両者の比較において良い方の値を太字で示している．また，ここにあげたすべてのアルゴリズムの中で最も良い値を示している値は，下線付きの太字で示した．

分類 i) に属する問題ではすべて *c*AS-TS が MMAS-TS よりも良い値を示し，また，*c*AS-TS はすべてのアルゴリズムの中でも最良となっている．なお，ここでは $\gamma = 0.4$ の場合の方が，$\gamma = 0.8$ の場合よりも全体として良い値を示している．分類 ii) に属する問題では，sko72 および sko81 を除く問題において *c*AS-TS が良い値を示している．なお，sko42 および sko49 では GH がすべてのアルゴリズムの中で最も良い値と

表 2 ローカルサーチを適用した場合の cAS の性能と他アルゴリズムの結果の比較．値（$Error$ 値（%））は 10 回の実験の平均値を示す．cAS-TS 以外のデータはいずれも文献 16) に記載された数値である

Table 2 Performance of cAS with local search on QAP. The results are average of $Error$ (%) over 10 independent runs.

分類	QAP	cAS-TS				MMAS-TS	MMAS-2OPT	HAS-QAP	GH	Ro-TS
		(γ=0.4)		(γ=0.8)						
		Error(%)	*Std*	*Error(%)*	*Std*	*Error(%)*				
i	tai35a	0.582	0.301	**0.572**	0.291	0.715	1.128	1.762	0.698	0.589
	tai40a	**0.726**	0.174	0.793	0.252	0.794	1.509	1.989	0.884	0.990
	tai50a	**1.051**	0.260	1.190	0.298	1.060	1.795	2.800	1.049	1.125
	tai60a	**1.059**	0.149	1.289	0.162	1.137	1.882	3.070	1.159	1.203
	tai80a	**0.740**	0.133	1.029	0.144	0.836	1.402	2.689	0.796	0.900
ii	sko42	**0.005**	0.011	0.008	0.016	0.032	0.051	0.076	**0.003**	0.025
	sko49	**0.044**	0.035	0.062	0.052	0.068	0.115	0.141	**0.040**	0.076
	sko56	**0.055**	0.081	0.065	0.049	0.075	0.098	0.101	0.060	0.088
	sko64	0.077	0.079	**0.035**	0.027	0.071	0.099	0.129	0.092	0.071
	sko72	0.126	0.074	0.091	0.051	**0.090**	0.172	0.277	0.143	0.146
	sko81	0.105	0.044	0.105	0.031	**0.062**	0.124	0.144	0.136	0.136
	sko90	**0.104**	0.047	0.168	0.008	0.114	0.140	0.231	0.196	0.128
iii	ste36a	0.139	0.217	0.118	0.114	**0.061**	0.126	n.a.	n.a.	0.155
	ste36b	**0**	0	**0**	0	**0**	**0**	n.a.	n.a.	0.081
iv	tai35b	0.098	0.168	**0**	0	0.051	**0**	0.026	0.107	0.064
	tai40b	0.403	0.848	**0**	0	0.402	**0**	0.000	0.211	0.531
	tai50b	0.183	0.190	**0.113**	0.156	0.172	**0.009**	0.192	0.214	0.342
	tai60b	0.280	0.217	0.091	0.156	**0.005**	**0.005**	0.048	0.291	0.417
	tai80b	0.716	0.394	**0.445**	0.418	0.591	**0.266**	0.667	0.829	1.031
	tai100b	0.263	0.152	**0.155**	0.105	0.230	**0.114**	n.a.	n.a.	0.512

Std: 標準偏差

なっているが，cAS-TS との差はわずかである．

分類 iii) に属する問題では，ste36a において MMAS-TS が cAS-TS より良い結果を示し，ste36b では両者ともに 0 である．分類 iv) に属する問題では，tai60b を除いて cAS-TS が MMAS-TS よりも良い結果を示している．なお，この分類の問題では最良の結果は MMAS-2OPT で得られており，分類 iv) に属する問題では，ローカルサーチとして 2OPT が優位であることが分かる．

以上，cAS-TS と MMAS-TS との比較を行い，またその他のアルゴリズムの結果を示した．cAS-TS と MMAS-TS との比較では，全 20 問題において 15 問題で cAS-TS が優れ，4 問題で MMAS-TS が優れた結果となり，全体として cAS-TS が良い結果を出している．なお，問題の分類によりローカルサーチの方法に優位差があり，分類 iv) に属する問題では 2OPT が優れた結果となっている．このように，アルゴリズム間で総実行時間を同一としてローカルサーチを併用する実験の結果から，cAS は MMAS と同等以上の性能を有しているといえる．しかし，ローカルサーチを用いない 4.1 節の結果ほどその差は顕著ではない．これはローカルサーチの影響の度合いが大きくなるためと思われる．なお，比較対象に用いた文献 16) のデータには平均値のみが記載されているので，統計的検定を行うことができないが，ここでは本研究の結果に関しては標準偏差（Std）を記載した．

4.3.2 γ の影響

図 4 で見たようにローカルサーチを適用しない場合は，γ が小さい値（$[0.1, 0.5]$）で良い性能を示した．ローカルサーチを適用した場合は，表 2 で見たように QAP の種別に依存して好ましい γ 値が異なっている．そこで，分類 i) から tai80a，分類 ii) から sko72，分類 iii) から ste36a，分類 iv) から tai80b を選び，図 4 の場合と同様，γ の値を 0.1 から 0.9 まで 0.1 刻みで変化させた場合の結果を**図 7** に示す．なお，表 2 の実験回数は，比較対象の論文の実験回数と同じく 10 回であったが，ここではデータの誤差を少なくするために実験回数は図 4 の場合と同じく 25 回とした．また，平均と標準偏差を用いて結果を示した．ここで，図 4 の場合と同様，各 γ 値に対して，中点は平均値の場所を示し，その上下の点は平均値から標準偏差の値だけずれた場所を示している．この図からも明らかなように，分類 i) の tai80a 以外の問題では γ 値が比較的大きいときに良い結果を示すことが分かる．これは，ローカルサーチを適用した実験では cAS の最大繰返し数を 50 としているので，小さい γ 値の場合，

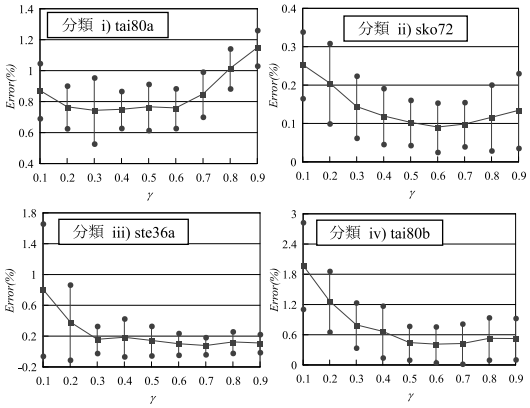

図 7 ローカルサーチを適用した場合の γ による $Error$ の変化
Fig. 7 $Error$ for various values of γ with local search.

収束抑制効果が強くなりすぎるためであると考えられる．このように，ローカルサーチを適用する場合には γ 値の適切な設定が重要となる．

5. *c*AS の並列化

進化的計算の並列化はよく知られた手法であり[3]，ACO アルゴリズムでも並列化の研究がある[10),14)]．並列化の目的は大きく 2 つに分類できる．1 つは，与えられた時間内に，よりクオリティの高い解を得ることである．他の 1 つは，与えられたクオリティの基準を満たす解を高速に得ることである．本章では，上記の第 2 の目的，すなわち高速化を達成することを目的に，複数のプロセッサを用いる *c*AS の並列化の一方法とその結果について述べる．

5.1 *c*AS の並列化のためのネットワーク基本モデル

本研究における *c*AS の実装には Java を用いている．Java には RMI や Jini[17] などネットワーク環境における分散処理のためのパッケージが準備されているが，本研究では *c*AS の並列化の基本アーキテクチャとして，クライアント/サーバモデルに基づくアプレットベースの基本モデルを構築した．この構造を**図 8** に示す．クライアントで実行されるプログラムはアプレットのクラスとしてサーバから転送される．すなわち，クライアントのプログラムはブラウザから Java アプレットとして実行される．

クライアントとサーバ間の通信は，TCP ソケットプログラミングを用い，両者の情報交換は Serializable インタフェースを実装したオブジェクトを交換して行う．具体的には ObjectInputStream，ObjectOutputStream オブジェクトの交換として通信する．このオブジェクトの詳細については Java 2 API[17] を参照

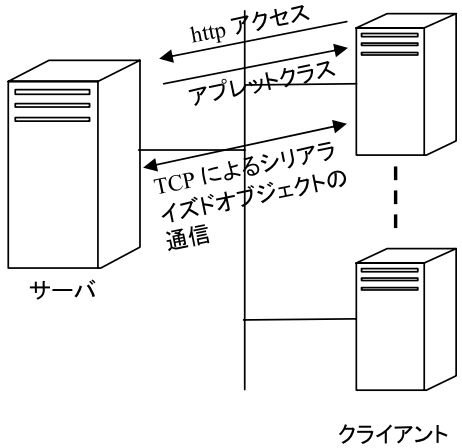

図 8 *c*AS の並列化のためのネットワーク基本モデル
Fig. 8 Network base model for parallelization of *c*AS.

表 3 ローカルサーチを用いた *c*AS における QAP での計算時間の分布．単位はミリ秒で，25 回の実験の平均値を示している
Table 3 Computational time of *c*AS with local search on QAP (in milliseconds).

QAP	サンプリング	ローカルサーチ	τ_{ij} の更新	その他	合計
tai40b	3.8	6833.8	1.3	12.3	6851.2
(%)	0.1%	99.7%	0.0%	0.2%	100.0%
tai50b	4.9	13465.8	2.3	13.3	13486.2
(%)	0.0%	99.8%	0.0%	0.1%	100.0%
tai60b	7.0	23448.7	3.0	15.7	23474.4
(%)	0.0%	99.9%	0.0%	0.1%	100.0%
tai80b	10.6	56688.7	5.9	19.9	56725.0
(%)	0.0%	99.9%	0.0%	0.0%	100.0%
tai100b	14.8	114411.7	9.0	21.4	114456.9
(%)	0.0%	100.0%	0.0%	0.0%	100.0%
tai150b	29.9	422079.8	19.9	46.1	422175.6
(%)	0.0%	100.0%	0.0%	0.0%	100.0%

していただきたい．

以上の方式により，Java アプレットの実行環境を有する任意のクライアントがサーバに http プロトコールによりアクセスすることで，簡単に並列計算環境を構築することができる．クライアント側で QAPLIB などのファイルのデータの参照が必要となる場合がある．本基本モデルでは，これらのファイルは Java の 1 つのクラスに変換しておくことを前提としている．これにより各クライアントで必要となるファイルのデータは http プロトコールによりクライアントからアクセスできる．なお，本基本モデルでは，サーバに Web サーバとしての機能を必要とする．

5.2 *p*-*c*AS による高速化方式の実装

表 3 は，4.3 節における実験において，$\gamma = 0.8$ としたときの *c*AS-TS の処理時間の分布を示したものである．処理時間の分布は問題の分類に依存しないので，ここでは分類 iv) に属する各サイズの問題を取り出している（tai150b（$n = 150$）は新たに追加）．こ

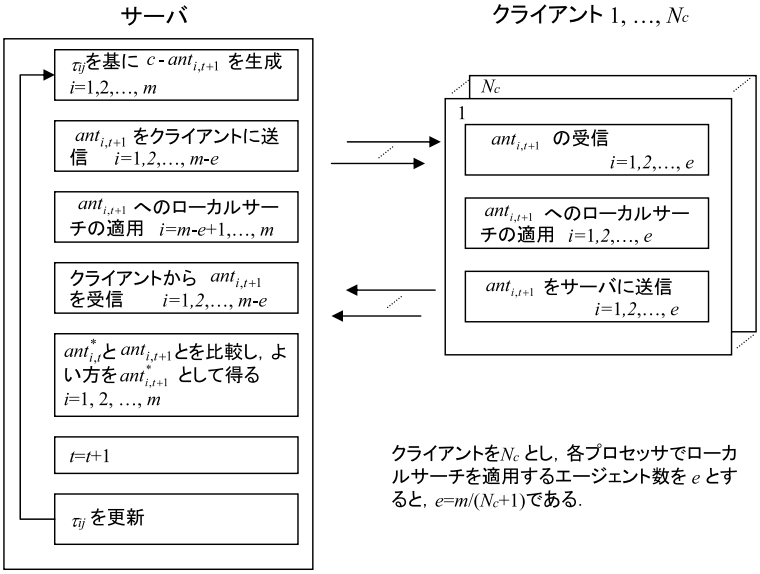

図 9 並列化 *c*AS（*p*-*c*AS）の概要
Fig. 9 Outline of the parallel *c*AS (*p*-*c*AS).

の表より，探索の総時間のほぼ 99%以上がローカルサーチ（Ro-TS）に費やされていることが分かる．そこで，今回の *p*-*c*AS では，ローカルサーチを並列化して高速化を図るという方式を 5.1 節で述べたネットワーク基本モデルを用いて実装した．

p-*c*AS の実行環境としては，4 章で述べた Opteron マシンを 2 台用い（以下，それぞれ，マシン A，マシン B と呼ぶ），両マシンを 1000BASE-T スイッチングハブで接続する．それぞれのマシンは 4 個のプロセッサを有する．マシン A をサーバとして用い，クライアント機能はマシン B に割り当てる．マシン A の Web サーバ用ソフトには Apache[1] を用いた．

図 9 は，上記の環境でのローカルサーチの並列化の方式である．ローカルサーチを適用して QAP を解くための *c*AS の論理的条件は 4.3 節と同じであり，$m = 5$ である．そこで，4 個の *c*-*ant* のローカルサーチをクライアントで行うこととし，このためマシン B には Java アプレットが実行できるブラウザを 4 つの独立したプロセスとして起動する．厳密には各ブラウザプロセスは OS により 4 つのプロセッサ間で動的に切り替わって実行されるが，これらのプロセスは 4 つのプロセッサに論理的に割り当てられると考えてよい．また，残りの 1 つの *c*-*ant* のローカルサーチはサーバプロセッサに割り当てる．なお，図 9 ではクライアントプロセッサ数を N_c として一般化して記述している．上記の説明は同図において，$N_c = 4$，$m = 5$ とした場合に相当し，ローカルサーチを適用するために

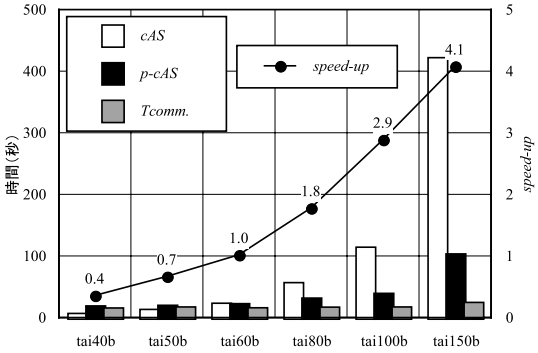

図 10 *p*-*c*AS による高速化の結果
Fig. 10 Performance of *p*-*c*AS and the speedups obtained on QAP.

各プロセッサに割り当てられるエージェントの数 e は 1 である．

5.3 *p*-*c*AS による高速化の効果

図 10 に結果をまとめる．論理的な実験条件は 4.3 節と同じである．データは 25 回の実験の平均である．ここで，*speed-up* は，(並列化しない場合の *c*AS の実行時間)/(*p*-*c*AS の実行時間) を示す．クライアントとサーバ間の通信オーバヘッド（$T_{comm.}$）がなければ *speed-up* の値は 5 に近い値となる．しかし同図に示しているように，通信オーバヘッド（$T_{comm.}$）が存在する．この通信オーバヘッドのために tai40b および tai50b では *speed-up* 値は 1 より小さく，また tai60b でほぼ 1 となっている．一方，より大きな問題である tai80b，tai100b および tai150b では，*speed-up* 値は

それぞれ，1.8，2.9 および 4.1 となっている．これは大きな問題では，全体の時間に占める $T_{comm.}$ の割合が小さくなるためであり，大きな問題での並列化による高速化への効果が大きいといえる．なお，この並列化方式では，より複雑なローカルサーチを用いる場合にはさらなる高速化の効果が得られる．

6. むすび

本論文では，cAS の QAP への適用方法について述べ，また実験によりその評価を主に MMAS との比較を通して行った．この結果，ローカルサーチを用いない場合の結果では，cAS は MMAS よりも優れた結果を示した．また総実行時間をアルゴリズム間で同一としてローカルサーチを用いる場合の結果でも，cAS-TS は MMAS-TS と同程度以上の性能を有し，また，他のメタヒューリスティックスよりも優れていることを示した．最後に，ローカルサーチを適用する cAS の高速化のための並列化の一手法についても述べ，その効果を確認した．

今後に残された問題としては，本研究で用いた実験条件と異なる条件での性能評価や 2OPT など他のローカルサーチの効果の検討などの研究がある．また，並列化手法に関しては，島モデルなど他の並列化のモデルについても検討していく必要がある．

謝辞 本論文の一部は，文部科学省科学研究補助金（基盤研究 C）課題番号 16500143 の一環として行われた．

参考文献

1) Apache Software Foundation: Apache HTTP Server Project. www.apache.org
2) Bullnheimer, B., Hartl, R.F. and Strauss, C.: A new rank based version of the Ant System: A computational study, *Cent. Europ. J. for Op. Research and Economics*, Vol.7, No.1, pp.25–38 (1999).
3) Cantu-Paz, E.: *Efficient and Accurate Parallel Genetic Algorithm*, Kluwer Academic Publishers, Boston (2000).
4) Dorigo, M. and Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE TEC*, Vol.1, No.1, pp.53–66 (1997).
5) Dorigo, M., Maniezzo, V. and Colorni, A.: The Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Trans. SMC-Part B*, Vol.26, No.1, pp.29–41 (1996).
6) Dorigo, M. and Stützle, T.: *Ant Colony Optimization*, MIT Press, Massachusetts (2004).
7) Taillard, É.D.: Robust taboo search for the quadratic assignment problem, *Parallel Computing*, Vol.17, pp.443–455 (1991).
8) Fleurent, C. and Ferland, J.A.: Genetic hybrids for the quadratic assignment problem, *Quadratic assignment and related problems*, Vol.16, pp.173–187 (1994).
9) Gambardella, L.M., Taillard, É.D. and Dorigo, M.: Ant colonies for the QAP, *Journal of the Operational Research Society*, Vol.50, pp.167–176 (1999).
10) Manfrin, M., Birattari, M., Stützle, T. and Dorigo, M.: Parallel Ant Colony Optimization for the Traveling Salesman Problems, *Proc. 5th Int. Workshop on Ant Colony Optimization and Swarm Intelligence* (*ANTS 2006*), pp.224–234 (2006).
11) Maniezzo, V.: Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, *INFORMS J. on Computing*, Vol.11, No.4, pp.358–369 (1999).
12) Maniezzo, V. and Colorini, A.: The Ant System applied to the quadratic assignment problem, *IEEE Trans. Data and Knowledge Engineering*, Vol.11, No.5, pp.769–778 (1999).
13) QAPLIB–A Quadratic Assignment Problem Library: www.opt.math.tu-graz.ac.at/qaplib
14) Randall, M. and Lewis, A.: A Parallel Implementation of Ant Colony Optimization, *Journal of Parallel and Distributed Computing*, Vol.62, No.9, pp.1421–1432 (2002).
15) Sahni, S. and Gonzalez, T.: P-complete approximation problems, *J. ACM*, Vol.23, pp.555–565 (1976).
16) Stützle, T. and Hoos, H.: Max-Min Ant System, *Future Generation Computer Systems*, Vol.16, No.9, pp.889–914 (2000).
17) Sun Microsystems, Inc.: Java 2 Platform, Standard Edition, v1.4.2 at API Specification. java.sun.com/j2se/1.4.2/docs/api
18) Taillard, É.D.: Robust tabu search implementation. mistic.heig-vd.ch/taillard
19) Tsutsui, S.: cAS: Ant Colony Optimization with Cunning Ants, *Proc. 9th Int. Conf. on Parallel Problem Solving from Nature* (*PPSN IX*), pp.162–171 (2006).
20) 新妻弘崇，石井 信，伊藤 実：アナログ λ-opt アルゴリズムを使った 2 次割当て問題の解法，電子情報通信学会論文誌 D-II，Vol.J82-D-II, No.12, pp.2375–2384 (1999).
21) 中道義之，有田隆也：ACO におけるランダム選択に基づく多様性調節の効果，情報処理学会論文誌，Vol.43, No.9, pp.2939–2947 (2003).

22) 筒井茂義：*c*AS:カニングアントを用いたACOの提案，人工知能学会論文誌，Vol.22, No.1, pp.29–36 (2007).
23) 又吉光邦，名嘉村盛和，宮城隼夫：改良2-opt法を組み込んだ遺伝的アルゴリズムによるQAPの解探索，電気学会論文誌C（電子･情報･システム部門誌），Vol.124-C, No.9, pp.1896–1906 (2004).

(平成19年 8 月 3 日受付)
(平成19年 9 月12日再受付)
(平成19年10月22日採録)

筒井　茂義（正会員）

1969年大阪市立大学大学院工学研究科修士課程修了．同年（株）日立製作所入社．中央研究所，システム開発研究所にて計算機システムの研究に従事．1987年より阪南大学に勤務．現在経営情報学部経営情報学科，兼大学院企業情報研究科教授．2000年4月～2001年3月イリノイ大学IlliGAL客員研究員．工学博士．進化的計算の研究に従事．1993年度人工知能学会全国大会優秀論文賞受賞，1996年度人工知能学会論文賞受賞．人工知能学会，IEEE各会員．

劉　　力綺

1982年生まれ．2004年逢甲大学工業工程学部卒業（台湾）．2006年4月阪南大学大学院修士課程企業情報研究科企業情報専攻入学．進化的計算を研究．スケジューリングシステムの研究に関心を持つ．

小島　基伸

1986年3月立命館大学理工学部卒業．同年（株）アセック入社．以後パソコンによるソフトウェア開発に従事．主に工場ラインのモニタおよびコントローラ，オフコンとの通信プログラムの開発．FA分野におけるファジー制御プログラムの開発．コンピュータ操作指導教育・コンサルティング（シグネット）．