様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成 25 年 3 月 31 日現在

機関番号：34425
研究種目：基盤研究(C)
研究期間：2010～2012
課題番号：22500215
研究課題名（和文）　アントコロニー最適化手法の枠組みの拡張に関する研究

研究課題名（英文）　Study on a new scheme for the ant colony optimization

研究代表者：筒井茂義（TSUTSUI SHIGEYOSHI）
阪南大学・経営情報学部・教授
研究者番号：90188590

研究成果の概要（和文）：アントコロニー最適化（Ant Colony Optimization, ACO）手法は，アリの採餌行動にヒントを得た探索手法であり，解くことが困難な多くの組合せ最適化問題に適用され，有効な結果が得られている．これらの問題を解く際には，高速にアルゴリズムを実行することが重要である．本研究では，(1) ACO をマルチコア計算機で高速に解く手法，(2) GPU (Graphic Processing Unit) により ACO を超並列計算する手法，の２つの提案を行った．ACO としては，申請者が提案した *c*AS (Cunning Ant System) を用いた．

研究成果の概要（英文）：As a bio-inspired computational paradigm, ant colony optimization (ACO) has been applied with great success to a large number of hard problems. In solving these problems, fast execution of the algorithm is important. In this research, we studied following two approaches for fast execution of the ACO algorithm; (1) parallel execution using multi-core processors, (2) massively parallel execution of the ACO algorithm using graphics processing units (GPUs). As for ACO algorithm, we used *c*AS (Cunning Ant System) which was developed in our previous study.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
2010 年度	1,100,000	330,000	1,430,000
2011 年度	900,000	270,000	1,170,000
2012 年度	800,000	240,000	1,040,000
総　計	2,800,000	840,000	3,640,000

研究分野：総合領域
科研費の分科・細目：情報学・感性情報学・ソフトコンピューティング
キーワード：アントコロニー最適化，マルチコア計算機，GPU，GPGPU，並列 ACO

１．研究開始当初の背景

アントコロニー最適化（Ant Colony Optimization, ACO）手法は，アリの採餌行動の際の経路生成過程にヒントを得た探索手法であり，巡回セールスマン問題（TSP）など多くの組合せ最適化問題に適用され，有効な結果が得られている．アリはフェロモンを介したコミュニケーションを行いながら群れで行動し，ある種の秩序を形成する．ACO では，この秩序形成過程を探索に用いる．ACO は，組合せ最適化問題など，解くことが困難な多くの問題に用いられている．これらの問題は実時間で高速に解くことが要求される場合が多くある．従って，ACO アルゴリ

ズムを高速に解くことが重要になっている．

２．研究の目的

ACO アルゴリズムの高速化の方法として，①マルチコアプロセッサによる並列実行の研究，② GPUs (Graphics Processing Units) を用いる超並列化の実装研究を行い，アルゴリズムの高速実効性を必要とする応用分野への ACO の適用の可能性を図る．

３．研究の方法

本研究では，ACO アルゴリズムとして，申請者が開発した *c*AS (Cunning Ant System)を用いた．巡回セールスマン問題 (Traveling Salesman Problem, TSP)および 2 次割当て問題 (Quadratic Assignment Problem, QAP)の 2 種類のベンチマーク問題を用いて，*c*AS のマルチコアプロセッサおよび GPU における並列化計算法の実装法を検討し，評価を行った．

４．研究成果

(1) *c*AS(Cunning Ant System)のレビュー：

本研究でベースとして用いる *c*AS は，TSP の解法を例として図 1 に示すように，アリ（以下，エージェント）が経路を生成する際に，既存解の一部（部分解）を借用し，残りの部分を通常の ACO アルゴリズムと同様フェロモン濃度に基づいて巡回路を生成する．部分解を提供するエージェントを *d-ant* (donor ant)，新しく生成されるエージェントを *c-ant* (cunning ant) と呼ぶ．

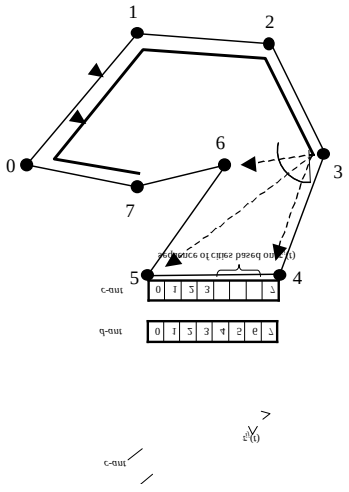

図 1　TSP の解法における *c-ant* と *d-ant*

生成された *c-ant* は，図 2 のコロニーモデルに示すように，*d-ant* と比較され，両者のうち，良い評価値（短い巡回路長）を持つものが集団（コロニー）に保持される．

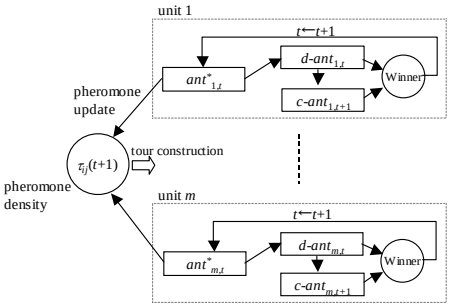

図 2　*c*AS におけるコロニーモデル

以上の戦略により，*c*AS は，exploration と exploitation とのバランスがとれた探索性能を有する ACO アルゴリズムとなっている．

(2) マルチコア計算機を用いた高速化並列 ACO：

今日の計算機は，複数の演算装置を有するマルチコア計算機が一般的になってきた．各コアは，主メモリを共有して並列に実行することが可能となっているので，並列処理に向いている．並列処理を実装する場合，性能は必ずしもコア数に比例しない．

式(1)は，アムダールの法則としてよく知られた式で，*n* 個のコアを用いた時の高速化の値 (Speedup 値) である．ここで，*p* はプログラムの中で並列化できる部分の時間の割合である．

$$Speedup = \frac{1}{(1-p)+\frac{p}{n}} \qquad (1)$$

p が 1 に近づけば Speedup 値は *n* に比例するが，一般には *p* は 1 未満である．この場合，*n* を無限大にしても Speedup 値は 1/(1-*p*) にしかならない．

進化計算は，集団を用いて多点探索することを特徴とする探索手法であるので，工夫により *p* を 1 に近づけることが可能なアルゴリズムである．本研究では，次の 3 つの ACO 並列化モデルを提案し，評価を行った．

(i) 同期型 ACO 並列化モデル（SP-ACO, Synchronous Parallel ACO）：フェロモンの更新は，各エージェントで同期して行う．その他の処理は，エージェント間ですべて並列処理とする．

(ii) 非同期型 ACO 並列化モデル（AP-ACO, Asynchronous Parallel ACO）：フェロモンの更新を，排他制御を用いて各エージェント間で非同期で行う．その他の処理は，エージェント間ですべて並列処理とする．

(iii) 粗非同期型 ACO 並列化モデル（RAP-ACO, Rough Asynchronous Parallel ACO）：(ii)と基本的に同じであるが，フェロ

モン更新の非同期制御で排他制御を行わない．排他制御による待ち時間はなくなるが，更新時の同時アクセスに伴う更新誤差が生じることを許す．

表 1 は，3 つの並列化による TSP の各インスタンスにおける実行時間（秒）と Speedup 値の結果である．ここではローカルサーチを用いない場合，3-opt を併用する場合，Lin-Kernighan (LK) を併用する場合を示している．なお，計算機は Intel Core i7 965（3.2GHz, 4 コア）である．

この結果では，ローカルサーチを用いない小さなテスト問題での Speedup 値においては，3 つの並列化に有意な差は見られない．しかし，3-opt および LK を併用した大きな問題においては，3 つの並列化方式に顕著な差がある．RAP-ACO は，3 つの並列化方式の中で大きな Speedup 値を示している．また，pcb442, att532, fl3795 では，RAP-ACO においてスーパーリニアを示している．RAP-ACO は，非同期方式であり，また排他制御もおこなっていないので，これらに伴う待ち時間がなく，また排他制御を行わないことでフェロモン更新にアクセス強豪に伴う部分的な更新誤差が生じる．これらが一種の突然変異的な効果が得られているものと考えられる．なお，表 1 には，統計検定の結果は省略している．

本研究により，マルチコア計算機における ACO の並列処理方式の比較と効果が確認できた．

表 1　マルチコア計算機による並列化の結果

Class	Instances	ACO	pararllel cAS (n_{core}=4)					
			SP-ACO		AP-ACO		RAP-ACO	
		T_{avg}	T_{avg}	*Speedup*	T_{avg}	*Speedup*	T_{avg}	*Speedup*
(1) no local search	berlin52	**0.22**	**0.10**	**2.3**	**0.09**	**2.5**	**0.08**	**2.7**
	st70	**1.46**	**0.57**	**2.6**	**0.57**	**2.6**	**0.47**	**3.1**
	pr76	**1.68**	**0.66**	**2.5**	**0.58**	**2.9**	**0.54**	**3.1**
(2) 3-OPT	pcb442	**10.15**	**2.50**	**4.1**	**3.31**	**3.1**	**2.01**	**5.1**
	att532	**22.05**	**9.11**	**2.4**	**8.20**	**2.7**	**5.38**	**4.1**
	rat783	**59.87**	**22.29**	**2.7**	**20.54**	**2.9**	**17.59**	**3.4**
(3) LK	pr2392	**55.26**	**19.06**	**2.9**	**17.17**	**3.2**	**18.71**	**3.0**
	fl3795	**227.47**	**60.77**	**3.7**	**58.83**	**3.9**	**49.02**	**4.6**
	rl5934	**736.55**	**197.65**	**3.7**	**189.85**	**3.9**	**180.33**	**4.1**

(3) GPUs (Graphics Processing Units) を用いる ACO の超並列化の実装研究：

①研究の背景

近年，パソコン(PC) の画面表示機能を担当する GPU（Graphics Processing Unit）がプログラミング可能になり，画像処理だけでなく一般的な計算プログラムも実行可能となった．最新の代表的な GPU である NVIDIA 社の GTX680 は 1536 個の小さいプロセッサを搭載し，数万の「スレッド」が同時実行可能な並列計算環境を低コストで提供している．そのピーク演算性能は 3TFLOPS（テラ・フロップス）を超える．このため GPU は価格性能比が非常によく，様々な科学技術計算へ GPU を適用する GPGPU (General Purpose computation on GPU) に関する研究がここ数年盛んになってきた．このように，PC に最新の GPU を付加することで，一昔前のスーパコンピュータに匹敵する計算機環境が構築できる．GPU では，同じ命令からなる処理が「スレッド」として多数生成され，それらが SIMD (Single Instruction, Multiple Data) 風に並列実行される．しかし，この並列実行では MIMD（Multiple Instruction, Multiple Data）として並列実行されるマルチコア CPU のように複数のスレッドを独立したプログラムとして柔軟に実行することができない．このため，並列計算に向いている進化計算といえども GPU の超並列性を効率的に実現するには，個別の問題に応じて進化計算のモデル構築や実装法にそれぞれ工夫が必要となる．

②研究の目的

本研究では，ACO を用いて，組合せ最適化問題の中で最も困難な問題の一つである 2 次割当て問題(Quadratic Assignment Problem, QAP)を GPGPU により超並列により高速に解く手法を確立する．

2 次割当て問題（QAP）は，n 個からなる部門を n 個の場所に，式(2) で定義される関数値が最小になるように割当てを決定する組合せ最適化問題である．

$$f(\phi) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_{ij} b_{\phi(i)\phi(j)} \qquad (2)$$

ここで，$A = (a_{ij})$ および $B = (b_{ij})$ はそれぞれ $n \times n$ のマトリックスであり，ϕ は $\{0, 1, \cdots, n-1\}$ の順列である．マトリックス A と B は，それぞれ，場所 i, j 間の距離，部門 i, j 間の流量（物流や人的交流の強さ）を表している．QAP は式(2) からもわかるように，評価関数が距離と流量との積になっているため，TSP に比べてはるかに解くことが困難であり，アルゴリズムの検証のベンチマーク問題にもよく用いられる．また，実際の応用問題も多くある．

QAP におけるローカルサーチとして 2-opt 法がよく知られている．一方，タブーサーチ (Tabu Search, TS)は，それ自体が組合せ最適化問題の解法に適用される強力なメタヒューリスティックスの一つであるが，進化計算と組合せてローカルサーチとしてもよく用いられる．本研究では，TS を ACO と組合せ，そのローカルサーチに用いる．

TS を QAP に適用する場合，現在の解のすべての近傍 $N(\phi)$ への移動による式(3) の変化量（以下，移動コストと呼ぶ）を計算しなければならない．近傍としては，ϕ の 2 つの位置の値を交換したものである．ϕ' を ϕ の r 番目の要素と s 番目の要素を交換して得られた近傍解とすると，移動コスト $\Delta(\phi, r, s) = f(\phi') - f(\phi)$ の計算量は，$O(n)$ となる．

ここで，もし$\Delta(\phi, r, s)$ の移動コストを記憶する $n \times n$ 個の記憶領域を用いると，ϕから u 番目の要素と v 番目の要素を交換して得られる近傍解への移動コストは，$\{u, v\} \cap \{r, s\} = \phi$が満たされる$\{u, v\}$ に対してはその計算量は O(1) となる．

このように，移動コストの計算量は，O(n)のものと O(1)のものとが混在する．

③GPU の概要

GPGPU に用いられる GPU は NVIDIA 社の GPU である．同社のアーキテクチャはほぼ 3 年単位に世代交代が行われており，2012 年になって Kepler アーキテクチャが発表された．本研究で用いる GPU は，Kepler の 1 世代前の Fermi アーキテクチャに基づく GTX 480 である（図 3 参照）．GPU 内におけるプロセッサは，CUDA core と呼ばれ，32 個の CUDA core が一つのグループとしてストリーミングマルチプロセッサ（Streaming Multiprocessor, SM）を構成している．GTX 480 は全体で 15 個の SM を有している．SM 内の各 CUDA core は，共有メモリ（shared memory）を介してデータを共有することができる．

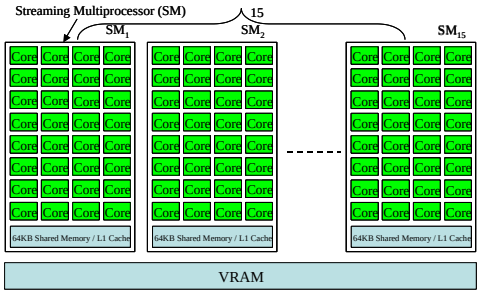

図 3　代表的な GPU（GTX480）

プログラムはスレッドとして実行される．ここで注意しなければならないことは，各スレッドは，「ウォープ(warp) 」と呼ばれる 32 スレッドを単位として，各 SP で SIMD (Single Instruction Multiple Data) 風に実行されることである．したがって，同一ウォープ内のスレッドの計算が，分岐などにより相互に大きく異なった処理を行う場合には，処理待ちに伴うアイドリングタイムが発生する．GPU におけるこのスレッドの実行法は，SIMT (Single Instruction, Multiple Threads）と呼ばれる．SIMT では，同一ウォープ内で，できるだけ分岐処理がないようにプログラムを作成し，アイドリングタイムを少なくすることが非常に重要となる．

④GPU による並列 ACO の構成

QAP を解くための並列化 ACO の全体構成を，図 4 に示す．この実装に当たっては，ACO の各ステップの機能は，すべて GPU で実行されるコード（カーネル関数）で実行される．CPU は ACO の各コードを順次呼出して ACO の繰返し制御を行うのみである．これにより，CPU と GPU との間では，探索の進行状況および最終解のデータのみを転送すればよい．

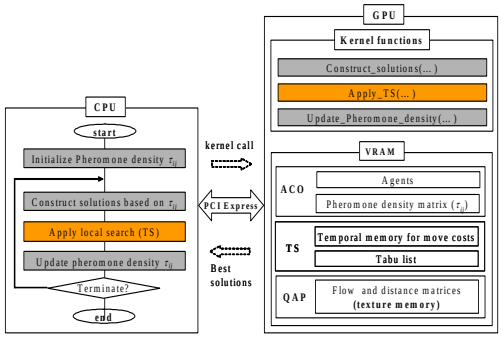

図 4　GPU による並列 ACO の構成

⑤移動コスト計算の効率的並列計算法の提案

GPU で並列実行されるプログラムは，スレッドとして実行されるが，スレッド間に分岐が存在すると③で述べたように，遅延が発生する．本解法では，TS を併用しているが，移動コストの計算量が全体の 99%以上占めている．移動コストの計算は，②で述べたように O(n)で計算できるものと O(1)で計算できるものとが混在している．このため，O(n)か O(1)かの分岐処理により遅延が起こり，並列処理の効率が低下する．本研究では MATA (Move-Cost Adjusted Thread Assignment)と呼ぶ手法を開発した．この原理を，図 5 に示す．

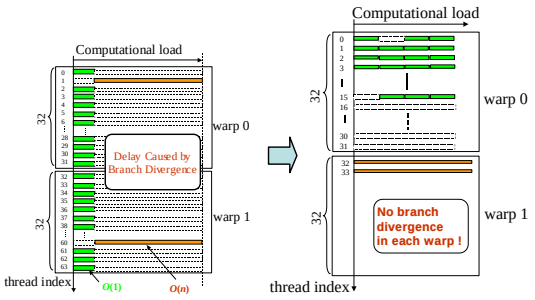

図 5　MATA による分岐レスプログラミング

⑥結果

QAP テスト問題を用いて実験した結果を表 2 に示す．この実験では，各問題に対して (1)MATA を用いた GPU 計算，(2)MATA を用いない，すなわち，ウォープ内で O(1) と O(n) の混在を許す GPU 計算（以下，non-MATA）および(3) シングルスレッドで行った CPU 計算の 3 種類の実験を行い，実行時間および最適解からの誤差(%) の 25 回の実行の平均を同表に示した．

まず，MATA と non-MATA の実行時間の平均（T_{avg}）の比を見ると，MATA の方が tai100a では 5.5 倍，tai100b では 5.5 倍高速

である．全体の平均で見ると 6.4 倍高速となっている．これは⑤で述べた MATA が有効であることを示している．このように，GPU 計算では問題の性質に応じて並列化の工夫が必要になる．本稿では言及しなかったが，メモリアクセスにおける遅延を最小化するような工夫も必要になる．次に GPGPU と CPU 計算の T_{avg} の比を MATA で見ると，GPU 計算は GPGPU に対して 16.9～35.5 倍高速になることを示しており，平均すると 24.2 倍の高速化が得られた．なお，比較に用いた CPU は，Core i7 965 (3.2 GHz)である．

表 2　GPU による並列計算の高速化の結果

QAP instances		GPU Computation (GTX 480)				CPU Computation		Speedup in T_{avg}
		T_{avg} (sec)			*Error* (%)	T_{avg} (sec)	*Error* (%)	CPU / MATA
		MATA	non-MATA	non-MATA / MATA				
class (i)	tai40a	9.2	70.8	7.7	0.15	191.4	0.13	20.8
	tai50a	17.8	126.6	7.1	0.36	464.0	0.35	26.1
	tai60a	34.8	283.9	8.2	0.36	963.8	0.36	27.7
	tai80a	153.9	728.6	4.7	0.38	3131.8	0.38	20.3
	tai100a	431.5	2357.3	5.5	0.35	7907.0	0.34	18.3
class (iv)	tai50b	0.2	1.5	6.3	**0**	6.0	**0**	24.9
	tai60b	0.4	2.8	7.7	**0**	12.7	**0**	35.5
	tai80b	6.6	33.9	5.1	**0**	141.6	**0**	21.4
	tai100b	10.1	55.2	5.5	**0**	296.5	**0**	29.5
	tai150b	2888.4	16348.8	5.7	0.05	48953.8	0.06	16.9
average		–	–	6.4	–	–	–	24.2

GPU による ACO の並列化の研究では，複数の GPU を用いる並列計算手法の研究も行い，ほぼ GPU 数に比例する高速化が得られた．

５．主な発表論文等

（研究代表者，研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 10 件）

①筒井 茂義，GPGPU を用いた並列アントコロニー最適化法による 2 次割当て問題の高速解法，電気学会論文誌 C（電子・情報・システム部門誌）,査読無（解説），Vol. 133, No. 3, 2013，583-595
DOI: 10.1541/ieejeiss.133.583

②Shigeyoshi Tsutsui，ACO on Multiple GPUs with CUDA for Faster Solution of QAPs, Proceedings of the Parallel Problem Solving from Nature - PPSN XII (PPSN 2012), Part II, 2012，査読有，174-184, Springer, Lecture Notes in Computer Science, Vol. 7492
URL: http://www.springer.com/computer/ai/book/978-3-642-32963-0

③Shigeyoshi Tsutsui and Noriyuki Fujimoto, On the effect of using multiple GPUs in solving QAPs with CUDA, Genetic and Evolutionary Computation Conference (GECCO 2012), Companion Material Proceedings, 査読有，629-630, 2012, ACM
URL: http://dl.acm.org/citation.cfm?id=2330893

④ Shigeyoshi Tsutsui and Noriyuki Fujimoto, Implementation of histogram based sampling algorithm within an EDA scheme with CUDA, Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2012), 1-8, 2012, IEEE
URL: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6256444&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F6241678%2F6252855%2F06256444.pdf%3Farnumber%3D6256444

⑤筒井 茂義，GPU を用いた高速並列進化計算による組合せ最適化問題へのアプローチ，オペレーションズ・リサーチ，査読無（解説），Vol. 57, No. 5, 2012, 261-269

⑥ Shigeyoshi Tsutsui and Lin Son, A Comparative Study of ACO and EDA Schemes in Solving QAPs, Proceedings of the 12th International Symposium on Advanced Intelligent System (ISIS 2011), 2011，査読無, 338-341

⑦Shigeyoshi Tsutsui and Noriyuki Fujimoto, ACO with Tabu Search on a GPU for Solving QAPs using Move-Cost Adjusted Thread Assignment, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011), 査読有，1547-1554, ACM, 2011
URL: http://dl.acm.org/citation.cfm?id=2001785&dl=ACM&coll=DL&CFID=214819452&CFTOKEN=31652249

⑧Shigeyoshi Tsutsui and Noriyuki Fujimoto, Fast QAP Solving by ACO with 2-opt Local Search on a GPU, Proceedings of the 2011 IEEE Congress on Evolutionary Computation (CEC-2011), 2011，査読有, 812-819, EEE
URL: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5949702&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F5936494%2F5949581%2F05949702.pdf%3Farnumber%3D5949702

⑨Shigeyoshi Tsutsui and Noriyuki Fujimoto, Parallel Ant Colony Optimization Algorithm on a Multi-core Processor, Proceedings of the 7-th International Conference on Swarm Intelligence (ANTS-2010), Vol. LNCS 6234, Lecture Notes in Computer Science, 2010，査読有, 488-495，Springer
URL: http://www.springer.com/computer/ai/book/978-3-642-15460-7?otherVersion=978-3-642-15461-4

⑩Noriyuki Fujimoto and Shigeyoshi Tsutsui, A Highly-Parallel TSP Solver for a GPU Computing Platform, Proceedings of the 7th International Conference on Numerical Methods and Applications, Vol. LNCS 6046, Lecture Notes in Computer Science, 2010，査読有, Springer, 264-271

URL: http://www.springer.com/computer/theoretical+computer+science/book/978-3-642-18465-9?otherVersion=978-3-642-18466-6

⑪Shigeyoshi Tsutsui and Noriyuki Fujimoto, An Analytical Study of GPU Computation for Solving QAPs by Parallel Evolutionary Computation with Independent Run, Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC-2010), 2010, 査読有, 889-896
URL: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?reload=true&arnumber=5585960

〔学会発表〕（計 7 件）

①筒井 茂義，進化計算による組合せ最適化問題へのアプローチ，日本オペレーションズ・リサーチ学会関西支部産学官交流会「インテリジェント技術と OR」，2011.11.26，大阪府立大学

②筒井 茂義，GPU 計算における SIMT を考慮した QAP の効率的な 並列計算解法について，平成 23 年（2011 年）電気学会 電子・情報・システム部門大会，2011.9.8，富山大学

③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，平成 23 年電気学会全国大会，2011.3.16-18，大阪大学

④筒井 茂義，藤本 典幸，GPU による並列 ACO を適用した QAP 高速解法について，進化計算学会 進化計算シンポジウム，2010.12.18，福岡レイクサイドホテル久山

④藤本 典幸，筒井 茂義，GPU を用いた高並列進化計算による巡回セールスマン問題の一解法，人工知能学会第 5 回進化計算フォロンティア研究会，2010.10.7-8，北海道大学

⑤藤本 典幸，筒井 茂義，CUDA GPU を用いた並列 GA による巡回セールスマン問題の解法，平成 22 年電気学会 電子・情報・システム部門大会，2010.9.2-3，熊本大学

⑥筒井 茂義，ACO アルゴリズムのマルチコア型計算機における並列化とその性能について，人工知能学会第 4 回進化計算フォロンティア研究会，2010.6.4-5，東京工業大学

〔図書〕（計 2 件）

①Shigeyoshi Tsutsui and Pierre Collet (Eds), Spriger, Massively Parallel Evolutionary Computation on GPGPUs, 2013, 418
URL: http://www.springer.com/computer/ai/book/978-3-642-37958-1.

②筒井 茂義，進化技術ハンドブック第 1 巻基礎編（13 章群知能の 13.1 節），近代科学社，2010，240

〔その他〕
ホームページ等
http://www.hannan-u.ac.jp/~tsutsui/

６．研究組織
(1)研究代表者
筒井 茂義（TSUTSUI SHIGEYOSHI）
阪南大学・経営情報学部・教授
研究者番号：90188590