社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－MPS－68 (20)
2008／3／5

アントコロニー最適化法の並列化手法および超並列SIMD型プロセッサへの実装

中野 光臣 †， 尼崎 太樹 †， 飯田 全広 †， 末吉 敏則 †

† 熊本大学大学院自然科学研究科
〒 860-8555 熊本県熊本市黒髪 2-39-1

近年，組合せ最適化問題を実時間で良質な解を得るために，近似解法の並列化やハードウェア化が行われている．著者らは，解の探索を独立して行うことが可能な組合せ最適化問題の近似解法であるアントコロニー最適化法 (ACO) に注目し，研究を行っている．本研究では，良解を得るために ACO の効果的な並列化を提案する．ACO の実装には，株式会社ルネサステクノロジが開発したマトリクス構造超並列プロセッサ（以下 MX コア）を用いた．MX コアは超並列細粒度 SIMD (Single Instruction Multiple Data) 型のプロセッサである．MX コアに ACO を並列化して実装し，TSP (Traveling Salesman Problem) を解くことで，解の精度と処理速度の評価を行う．その結果，細粒度方式を用いた並列化により Pentium M と比較して解の精度が 0.5% 良くなった．

A parallelization technique of ACO, and implement to massively parallel SIMD processor

Mitsutaka Nakano † Motoki Amagasaki † Masahiro Iida † Toshinori Sueyoshi †

† Graduate School of Science and Technology, Kumamoto University 2-39-1 Kurokami, Kumamoto-shi, 860-8555 Japan

In the combinationial optimization problem, parallelized and hardware implementation of approximate means are performed. We study Ant Colony Optimization (ACO) which is heuristic algorithm for combinatonial optimization problem. We propose effective parallelized implementation of ACO which obtain a good precision. We implemented ACO to massively parallel processor based on the matrix architecture (MX Core). MX Core realizes highly parallel processing. In this paper, we focus on the parallel processing and discuss to implement an ACO for MX Core. We evaluate ACO by solving Traveling Salesman Problem (TSP) on MX Core. As a result, the accuracy of the solution has improved 0.5% compared with Pentium M.

1 はじめに

現在，組合せ最適化問題に対して実時間で良質な解を得るために，近似解法の利用や処理の並列化，専用回路への実装など処理を高速化する様々な研究が行われている．最適化問題の近似解法の1つとして，アントコロニー最適化法（ACO）[1] がある．ACO はマルチエージェント型の解法であり，エージェントごとの探索結果が独立して得られる．そのため，処理の並列度を高めることが可能である．我々は，最適化問題を解く上でこれらの特徴を持つ ACO に注目している．

本研究では，ACO を効率よく並列化することで，組合せ最適化問題の良質な解を高速に得ることを目的としている．我々は，ACO の効果的な実装を行うため，株式会社ルネサステクノロジ社が開発したマトリクス構造超並列プロセッサコア（以下 MX コア）[2] を用いた．MX コアは，超並列細粒度 SIMD (Single Instruction Multiple Data) 型のプロセッサである．本稿では，アントコロニー最適化法の探索処理の並列化手法を提案し，MX コアへ実装を行う．また，巡回セールスマン問題を用いて評価を行う．

以下，2 章で ACO について述べ，3 章では MX コアの特徴と構造を紹介する．4 章では，ACO の並列化手法を示し，5 章で評価環境と評価結果を示す．最後に 6 章でまとめと今後の課題を述べる．

2 アントコロニー最適化法

ACOは，アリが食料採集を行う際，最短経路を通ることを模倣した最適化手法である．

2.1 Max-Min Ant System

本研究では，ACOの一手法であるMax-Min Ant System（MMAS）[3]を採用する．ACOは，まず複数のエージェント（ant）がそれぞれ解を探索し，1つの解を生成する．各エージェントは独立して動作するため，並列に処理を行うことが可能である．その後，各エージェントの探索結果をまとめ，評価を行う．結果はフェロモン情報としてフェロモンの更新時に記録される．

次に，フェロモンの更新処理を行う．MMASにおける更新処理は，都市iと都市j間のフェロモン値$\tau(i,j)$に対して，上限値τ_{MAX}と下限値τ_{MIN}が設定される．フェロモンの増加量は式(1)に従う．ρはフェロモンの蒸発率を表し，本稿では0.5とおく．$\Delta\tau(i,j)$の値は式2で決定される．T^*はアルゴリズムの各ステップにおいて生成された巡回路のうち，総経路長が最短のものであり，L^*はその総経路長を表す．したがって，各ステップで最も良い解を生成したエージェントが用いた経路のみに適用される．それぞれの経路に対して式1を用いてフェロモンの更新を行ったあと，更新されたフェロモンを上限値τ_{MAX}と下限値τ_{MIN}間に限定させる標準化が行われる．

$$\tau(i,j) = (1-\rho)\tau(i,j) + \rho\Delta\tau(i,j) \quad (1)$$

$$\Delta\tau(i,j) = \begin{cases} 1/L^* & \text{if } (i,j) \in T^* \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

最後に，探索結果と前回の最良解を比較し最良解を更新する．ここまでを1ステップとする．ステップが繰り返されることで情報がより洗練される．また，良い解の情報をもとに探索が行われるため，解の質も向上する．

3 マトリクス構造超並列プロセッサコア

本章では，MXコアの構造および処理機能，MXコアを組み込んだシステム構成について述べる．

3.1 MXコアの構造

図1に示すように，MXコアは主にPE (Processing Element)とデータレジスタで構成される．データレジスタは，PEの両翼に512bitづつのSRAMが配置される．これをMXコアの基本構成単位とし，PEと1,024bitのデータレジスタをあわせてエントリと呼ぶ．MXコア全体でPEは1,024エントリあり，このPEをSIMD命令で実行することにより，最大1,024個のデータに対し並列処理を行う．MXコアは一つのチップ上にホストCPUとなるメインプロセッサおよびDMAC (Direct Memory Access Controller)，メモリコントローラ，SDRAMと合わせて実装され，さまざまなアプリケーションにおいてデータ処理アクセラレータとして機能する．組込みシステム向けアクセラレータであるため，小面積・低消費電力を実現している．

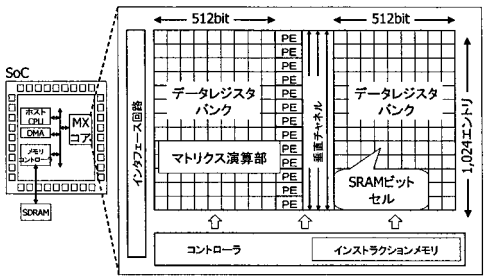

図1 MXコアの構成

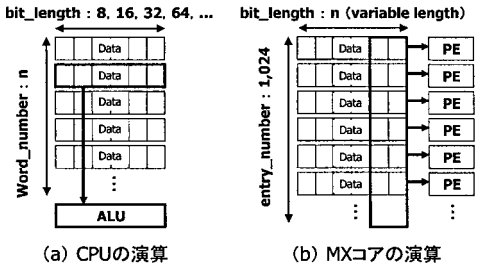

図2 演算方法

3.2 MXコアの演算方法の特徴

図2(a)のように，汎用プロセッサの演算は逐次的にデータの数だけ繰り返す必要がある．一方，MXコアは図2(b)に示すように，1,024個のデータを同時に2bitずつ演算を行う．このため，1つのデータに対しては複数サイクルを要するが，超並列構造を生かし1,024個のデータを同時に処理することにより，演算サイクルのコストを小さくできる．また，PEの処理幅が2bitという細粒度構造であるために，データ幅の制約が無く様々なbit幅のデータに対応可能となる．

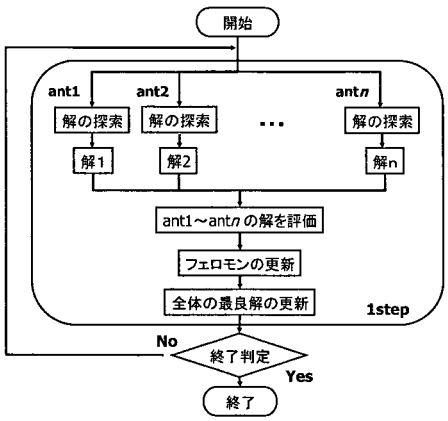

図 3　ACO の動作フロー

Entry	Data register 0				Data register 1	
0	ant0	City_data 0-4	tmp	PE	City_data 5-9	tmp
1		City_data 10-14		PE	City_data 15-19	
2		City_data 20-24		PE	City_data 25-29	
3		City_data 30-34		PE	City_data 35-39	
4		City_data 40-44		PE	City_data 45-49	
5				PE		
6	ant1	City_data 0-4		PE	City_data 0-4	
7		City_data 10-14		PE	City_data 10-14	
⋮		⋮		⋮	⋮	
1022				PE		
1023				PE		

図 4　探索部の MX コアへのデータ配置

3.3　浮動小数点演算

MX コアは FPU (Floating Point Processing Unit) を持たないので，浮動小数点演算を算術論理演算の組合せにより実現してきた[4]．本稿においても浮動小数点数は，上記の方法で IEEE754 形式に準拠した 32bit 単精度浮動小数点数を用いる．

4　ACO の並列化手法と実装

4.1　ACO の動作フロー

ACO の動作フローを図 3 に示す．ACO は，フリーで提供されている TSP を解くプログラム[5]を参考にし，実装を行う．MX コアでは，並列化が抽出できる解の探索処理および解の導出，フェロモン情報の更新を実装する．解の評価および解の更新は，並列性が出ないので CPU で処理を行う．

4.2　探索処理の並列化

解の探索処理の並列化ついて述べる．各エントリをエージェントとみなし，MX コアへの入力は距離の短い順にデータを並べて入力を行う．都市数やフェロモン情報による制約はあるが，エージェントの数だけ並列演算が可能である．MX コアへの初期データ配置を図 4 に示す．この図は，MX コアのマトリクス演算部を模式的に表した図である．なお，図 4 では TSPLIB[6] に含まれる eil51 を一例としている．

図 4 の Citydata とは，その都市への経路上のフェロモン情報を意味し，それに基づいて経路探索を行う．処理に使用するフェロモン情報は，32bit 単精度浮動小数点数を用いる．テンポラリ領域等を考慮し，1 エントリに 10 都市分のデータを配置する．n 都市問題において，入力データは現在エージェントが居る都市データを除く $n-1$ 個であり，1 エージェントあたり $\frac{n-1}{10}$ エントリが使用される．探索処理を行ううえで，まず各エントリの 10 都市で探索を行う．各エージェント $\frac{n-1}{10}$ 並列で探索処理を行うことが可能である．その後，各エントリで導出された都市に対しもう一度探索処理を行う．2 度の探索処理を行い，$n-1$ 都市から 1 都市を導出する．この手法は，ACO をエージェント数で並列化し，さらに細粒度演算によって探索部分も並列処理している．

4.3　フェロモンの更新

フェロモン情報の更新処理は式 (1) を用いる．全フェロモンが独立して同じ処理を行うため，最大 1,024 並列の処理が可能である．比較処理も同様である．

5　評価

MX コアの評価モデルは，ルネサス社提供の方式シミュレータ version 0.05.00 ASIM を用いて計測する．MX コアの動作周波数を 200MHz と設定し，計測時間を評価している．また，比較対象は Intel 社 Pentium M 1.1GHz とし，前述の ACO プログラムの処理を変えずに使用している．本研究において評価に使用した TSP は，TSPLIB にある 51 都市問題（eil51，最適解 426），100 都市問題（kroA100，最適解 21,282）である．

5.1　エージェント数変化による解の収束

MX コアに ACO を実装し，100 ステップ繰り返した時の導出した解について，最適解との誤差を図 5 に示す．縦軸は，最適解からの誤差を示している．横軸はエージェント数を示している．それぞれの解の値は，10 回試行した結果について平均

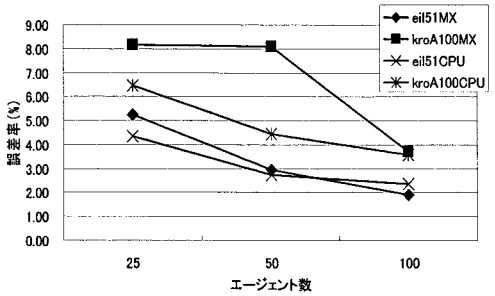

図 5　各 TSP における解の誤差

表 1　100 ステップに要する処理時間 [sec]

	問題	エージェント数		
		25	50	100
MX コア 200MHz	eil51	1.49	1.49	1.49
	kroA100	3.77	3.77	3.77
Pentium M 1.1GHz	eil51	0.13	0.20	0.34
	kroA100	0.32	0.50	0.82

を取ったものである．グラフの MX は MX コアで処理した結果，CPU は比較プロセッサで処理した結果である．

この結果から，探索の並列度が上がることにより，導出される解の精度が良くなっていることがわかる．すべての問題で，エージェント数が 25 より 100 のほうが 4% 程度精度が上がっている．また，エージェント数が少ないと提案手法の精度が比較対象より悪いが，エージェント数が増えると，精度が比較対象より良くなっていることがわかる．エージェント数の増加により，さらに精度が向上すると考えられる．

5.2　エージェント数変化による処理時間

表 1 は，100 ステップ処理を行ったときの処理時間を，エージェント数ごとに示したものである．Pentium M と比較すると，MX コアが約 4 倍から 10 倍程度遅い．これは，実装した処理に浮動小数点演算が多く含まれているため，FPU を持たない MX コアの処理が重くなっていると考える．しかし，MX コアでは並列化のため処理時間が一定であるが，Pentium M は線形に増加している．MX コアは小面積，低消費電力であるので複数個組み合わせて並列度を上げても十分に実用的である [2]．この結果から，エージェント数が増加すると Pentium M より高速に処理できると考えられる．また，MX コアにおいて，処理における演算精度が確保できるならば，処理データ幅を削減し処理速度を上げることも可能である．

6　まとめと今後の課題

本稿では，組合せ最適化問題の近似解法であるアントコロニー最適化法の並列化手法を提案し，MX コアへ実装した．結果より，細粒度演算による探索処理の並列化および 100 エージェントの並列度で解の探索処理を行うことができ，0.5% 程度精度が良くなった．よって，MX コアの超並列演算を使用し，ACO の並列化手法の有効性を確認することができた．今後の課題として，実装において MX コアと外部とのデータの入出力は考慮に入れていないので，入出力を考慮したデータ入出力が必要となる．また，MX コアの細粒度演算を生かした演算処理の速度向上を考える必要がある．

参考文献

1) 大内東，山本雅人，川村秀憲，柴肇一，高柳俊明，當間愛晃，遠藤聡志．**“生命複雑系からの計算パラダイム—アントコロニー最適化法・*DNA* コンピューティング・免疫システム.”** 森北出版, 2003.
2) K.Mizumoto et al.: *“A multi matrix-processor core architecture for real-time image processing Soc,”* A-SSCC, no.6-2, pp. 180-183, Jeju, Korea, Nov., 2006.
3) Thomas Stutzle and Holger Hoos: *“MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems,”* Metaheuristics: Advance and Trends in Local Search Paradigms for Optimization, pages 137-154, 1998.
4) 山崎博之，飯田全広，水本勝也，山本治，末吉敏則：**“単純な *SIMD* 演算の組み合せによる高速実数演算の実現,”** 信学技法 RECONF2005-23(2005-5), Vol.105, No.43, pp.49-54, May. 2005.
5) Ant Colony Optimization http://iridia.ulb.ac.be/ mdorigo/ ACO/ACO.html
6) Traveling Salesman Problem http://www.iwr.uni-heidelberg.de/groups/ comopt/software/TSPLIB95/index.html