Cell Broadband EngineによるPSO演算の実験、検証

藤田　大地†　三浦　康之†　渡辺　重佳†
湘南工科大学†

1.はじめに

粒子群最適化(ParticleSwarmOptimization, PSO)は医術や映画などと様々な場面で応用されている汎用的な最適化アルゴリズムの一つであり、遺伝的アルゴリズム(GeneticAlgorithm, GA)などと比べて収束が早いことが知られている。PSOは、多大な計算コストが必要となるものの、並列化に適したアルゴリズムであることから、マルチコアプロセッサなどを用いて効率的に並列化を行うことが可能と考えられる。

またこの分野に関して過去に様々な成果が発表されており、CellB.E.に関してはSIMD演算とSPEの活用により、高い効率で並列化できることが知られている[1]。しかしながら局所的な最適値を扱うものに関しては問題が複雑なことから、並列化に関する研究が少ないのが現状である。

本稿ではローカルな最適値をCell上で演算するうえでのアルゴリズムの検討と前準備としてグローバルな最適値を複数のSPUに実装、検証した。

2.粒子群最適化

PSOは群知能の一種である。群れの探索点において、それぞれが情報を共有しながら最適な解を探索する。

各探索点は、位置ベクトル、速度ベクトル、各探索点の過去の最良値とその時の位置ベクトルを保持する。それと共に、全体では過去の最適値とその時の位置ベクトルの情報を共有する。

以下にPSOの処理手順を示す

1．各粒子について位置ベクトルxと速度ベクトルvをランダムに初期化する。
2．各粒子の最適値を現在位置に初期化する。
3．全体の最適値を各粒子が算出した最適値に置き換える
4．各探索点の位置と速度を以下の数式によって演算する。

$x = x + v$…①

$v=\alpha v+\beta_1\omega_1(x'-x)+\beta_2\omega_2(x'_g-x)$…②

ここでα, β_1, β_2は定数で一般的には1に近い値が望ましい。ω_1, ω_2は1～0の範囲の乱数である。

5．x′とx'_g、vを更新する
x'は各粒子が過去に発見したベストの値である。x'gは全体が過去に発見したベストの値である。
6．3～5の手順を繰り返す。

3.Cell Broadband Engine

PlayStation 3 (PS3®)に搭載されているCell（Cell Broadband Engine）はソニー・SCE・IBM・東芝によって開発されたマイクロプロセッサである．Cellには PowerPC ベースの CPU である PPE が1基，SIMD 演算機である SPE が8基搭載されている．2 つはそれぞれ得意とする分野が違い，有効に活用することで Cell の能力を発揮することができる

4.環境構築

4.1 LINUX環境の構築

PS3®には PowerPC ベースの CPU が搭載されており，LINUX をインストールすることができる．我々の研究室では Fedora 11 を導入している．

4.2 Cellプログラミング環境の構築

プログラミング環境として，libspe2 を利用する．Libspe2 は，(株)ソニー・コンピュータエンタテインメント，IBM 社及び(株)東芝の 3 社によって共同で開発され，GNU LGPL に基づき公開されている．Cell の SPE を扱うために必要なヘッダーファイルなどが含まれる．

5.並列化

5.1 グローバルな最適値によるアルゴリズムの並列化

グローバルな最適値によるアルゴリズムの並列化の流れを図1に示す。複数のSPUへの対応としてSPU/エージェント数を各SPUに割り当てた。各SPUからの最適値をまとめるためにはデータを1サイクルごとに全体に通信する必要がある。

Experiment and verification of Particle Swarm Optimization by Cell Broadband Engine
†Daichi.Fujita †Yasuyuki Miura †Shigeyosi Watanabe
†Shonan Institute of Technology

CellB.E.には PPE と各 SPE との通信手段として DMA 通信と mailbox がある。今回扱ったデータ量は 32bit 以下であったため後者により PPE に各 SPU からの最適値を集中させた。同 PPE により最適値を計算したものを再び mailbox により各 SPU に返した。

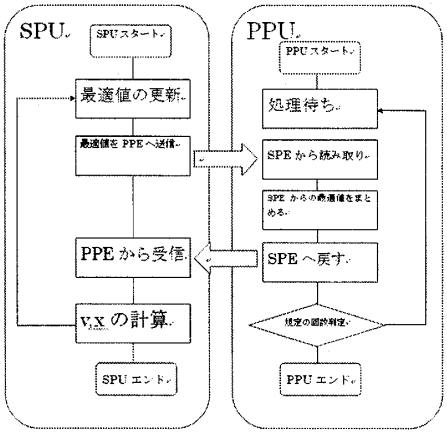

図 1. グローバル最適値フローチャート

5.2 ローカル最適値によるアルゴリズムの並列化

ローカル最適値によるアルゴリズムの並列化の流れを図 2 に示す。各エージェントの初期配置として、SPE ごとに多次元空間を分割して配置する。空間分割の場合、境界の近傍にあるエージェントに対し SPU 間で通信が発生する可能性がある。この場合通信頻度を少なくするためにエージェント同士を交換する方法が適切であると思われる。エージェント同士を効果的に交換することにより SPE 同士のロードバランスを維持しつつ、SPE 間の通信回数を抑えることができると考えられる。

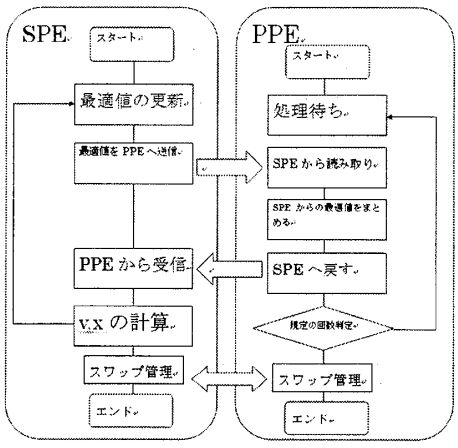

図 2．ローカル最適値フローチャート

6.実験結果

実験環境として PLAY STAION3 (40GB) を使用している。これは CPU である Cell B.E.3.2GHz を搭載しており、メモリは 256MB である。OS は Fedora 11、コンパイラは GCC4.1.1 を使用している。SPU を使うためのツール群として CellSDK3.1、libspe2 を使用している。

表 1．グローバルな最適値

Iter	Agen	SPU 数→	1	2	4
1000	1024	計算時間(s)	5.8	2.93	1.52
		速度向上率		1.98	3.81
	128	計算時間(s)	0.75	0.41	0.25
		速度向上率		1.82	2.98
100	1024	計算時間(s)	0.6	0.31	0.18
		速度向上率		1.9	3.34
	128	計算時間(s)	0.09	0.06	0.05
		速度向上率		1.54	1.78

Cell 上でグローバルな最適値によるアルゴリズムの並列化を施した PSO の計算を行ったときの性能を調べた。②式のα,β_1,β_2はすべて 0.9 としている。

表 1 は反復(iter)とエージェント数(Agen)を変動させたときの SPU のコア数による速度向上率を示したものである。

全てのケースで高速に計算されている。「iter=1000、Agen=1024」の 4 基では 3.8 倍以上高速に計算された。「iter=100、Agen=128」の 4 基では 2 倍以下と非常に遅い結果となった。これは演算時間に比べ SPU 間の通信などの時間が相対的に多くなるためである。

7.まとめ

本稿では、CellB.E.上で PSO を並列に実行するアルゴリズムを提案した。5.1 のアルゴリズムを検証した結果、最大 3.8 倍の性能を示した。今後は 5.2 によるアルゴリズムを検証し、SIMD などの高速化技法を適応していく。

参考文献

[1] 五十嵐潤, 園尾聡, 古賀崇了, “Cell B.E. による SIMD-oriented Fast Mersenne Twister を用いた Particle Swarm Optimization, ” Cell Speed Challenge 2008 自由課題部門, 2008.6

[2] PLAYSTATION 3 Linux 完全攻略ガイド 著者：塩田紳二, 安田絹子, 國司光宣, 平初, 川井義治, 古坂大地, 青柳信吾, 八重樫剛史 ISBN コード：978-4-8443-23891

[3] http://cell.fixstars.com