Vol.2012-MPS-87 No.18
2012/3/2

マルチコアCPUにおける並行型差分進化のノンパラメトリック検定を用いた比較研究

田　川　聖　治[†1]

マルチコア CPU を対象とした差分進化の並行プログラムである並行型差分進化を紹介する．次に，2 種類のマルチコア CPU において，並行型差分進化の性能を評価する．数値実験とノンパラメトリック検定から，マルチコア CPU の種類とスレッド数が，並行型差分進化の実行時間のみならず，解の精度にも影響することを示す．

A Study of Concurrent Differential Evolution on Multi-core CPUs by using Non-parametric Test

Kiyoharu Tagawa[†1]

In order to use multi-core CPUs effectively, a concurrent program of Differential Evolution (DE) called Concurrent DE (CDE) is described. The performances of CDE are compared between two kinds of popular multi-core CPUs. Through numerical experiments and non-parametric tests, it is shown that not only the execution time of CDE but also the quality of solution obtained by CDE depends on the kinds of multi-core CPUs and the number of threads.

1. はじめに

近年，1つのチップ上に複数のコア（プロセッサ）を集積したマルチコア CPU がパソコンにおいて広く普及している．このため，複数のコアによる並列処理を行うことで，様々なアプリケーションを高速化できる並行プログラムが注目されている[1)]．本稿では，マルチコア CPU を対象とした差分進化（DE：Differential Evolution）[2)] の並行プログラムである並行型差分進化（CDE：Concurrent DE）[3)] について紹介する．次に，代表的なマルチコア CPU である Intel 社の Core アーキテクチャと AMD 社の Phenom アーキテクチャにおいて，CDE の性能を比較する．数値実験とノンパラメトリック検定[4)] から，マルチコア CPU のアーキテクチャの違いと CDE から呼び出されるワーカ（スレッド）数が，CDE の実行時間のみならず，CDE によって得られる解の精度にも影響することを明らかにする．

†1 近畿大学
Kinki University

2. 並行型差分進化（CDE）

2.1 個体表現

関数最適化問題の最適解は目的関数 $f(\boldsymbol{x})$ を最小とする D 個の決定変数 $x_j \in \mathbb{R}$ である．CDE では関数最適化問題の解候補を個体とし，個体の配列を集団とする．すなわち，個体数 N_P の集団 $\mathbf{P}$ において i 番目の個体 $\boldsymbol{x}_i \in \mathbf{P}$ は，式 (1) の実数ベクトルとなる．

$$\boldsymbol{x}_i = (x_{1,i}, \cdots, x_{j,i}, \cdots, x_{D,i}),\ i = 1, \cdots, N_P \tag{1}$$

2.2 個体の生成

CDE ではターゲットベクトルと呼ばれる親個体 $\boldsymbol{x}_i \in \mathbf{P}$ を順番に指定し，各 $\boldsymbol{x}_i$ に DE の戦略[2)] と呼ばれる遺伝的操作を適用することで，新たな個体の候補であるトライアルベクトル $\boldsymbol{u}$ を生成する．本稿で使用した最も基本的な DE の戦略「DE/rand/1/exp」[2)] では，集団 $\mathbf{P}$ からランダムに 3 つの異なる親個体 $\boldsymbol{x}_{r1}$，$\boldsymbol{x}_{r2}$，$\boldsymbol{x}_{r3}$ （$i \neq r1 \neq r2 \neq r3$）を選択し，スケール係数 F に基づく式 (2) の差分突然変異により変異ベクトル $\boldsymbol{v}$ を生成する．

$$\boldsymbol{v} = \boldsymbol{x}_{r1} + F\left(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}\right) \tag{2}$$

次に，ターゲットベクトル $\boldsymbol{x}_i$ と $\boldsymbol{v}$ の指数交叉により，トライアルベクトル $\boldsymbol{u}$ を生成する．指数交叉では交叉率 C_R に基づき $\boldsymbol{x}_i$ と $\boldsymbol{v}$ から要素を確率的に選択して $\boldsymbol{u}$ の要素とする．

2.3 集団の更新

集団 $\mathbf{P}$ を式 (3) のように N_T 個の部分集団 $\mathbf{P}_n$ に分割し，$\boldsymbol{x}_i \in \mathbf{P}_n$ の処理をワーカ W_n に割当てる．各ワーカはスレッドで実装し，集団 $\mathbf{P}$ へのアクセスは並行参照と排他更新[1)] を想定する．個体 $\boldsymbol{x}_i \in \mathbf{P}_n$ はワーカ W_n のみが更新するため，排他制御は不要である．

$$\mathbf{P} = \mathbf{P}_1 \cup \cdots \cup \mathbf{P}_n \cup \cdots \cup \mathbf{P}_{N_T} \tag{3}$$

〔CDE のメインルーチンの手順〕

手順 1: 個体 $\boldsymbol{x}_i \in \mathbf{P}$（$i = 1, \cdots, N_P$）をランダムに生成する．

手順 2: ワーカ W_n（$n = 1, \cdots, N_T$）を呼び出す．

手順 3: 全ワーカ W_n が終了するまで待つ．

手順 4: 最良の個体 $\boldsymbol{x}_b \in \mathbf{P}$ を出力して終了する．

表 1 実行時間の比較 [ms]
Table 1 Comparison of execution times [ms]

N_T	1	2	4	6	8
CPU	Intel(R) Core(TM) i7				
f_1	142.73 (5.41)	100.00 (10.53)	60.43 (7.94)	51.56 (8.18)	**45.83** (6.96)
f_2	357.30 (9.89)	196.33 (8.94)	106.26 (10.52)	90.63 (7.61)	**74.46** (8.82)
CPU	AMD Phenom(TM) II X6				
f_1	194.63 (8.95)	170.70 (19.36)	95.76 (11.38)	70.26 (8.01)	**65.03** (7.17)
f_2	397.03 (10.50)	258.80 (10.99)	141.03 (6.44)	**101.96** (8.92)	106.20 (8.66)

表 2 目的関数値の比較
Table 2 Comparison of objective function values

N_T	1	2	4	6	8
CPU	Intel(R) Core(TM) i7				
f_1	3.58E-10 (9.10E-11)	**3.02E-10** (9.10E-11)	6.57E-10 (4.56E-10)	8.81E-10 (6.10E-10)	7.50E-10 (4.80E-10)
f_2	7.16E-9 (8.09E-9)	6.11E-8 (2.41E-7)	**4.11E-9** (2.95E-9)	3.46E-8 (9.26E-8)	4.56E-9 (5.71E-9)
CPU	AMD Phenom(TM) II X6				
f_1	3.58E-10 (9.10E-11)	**3.36E-10** (9.45E-11)	3.97E-9 (3.30E-9)	2.96E-9 (6.31E-9)	2.86E-7 (2.09E-7)
f_2	**7.16E-9** (8.09E-9)	2.57E-8 (6.10E-8)	2.17E-8 (4.88E-8)	1.07E-8 (2.16E-8)	3.03E-6 (4.44E-6)

表 3 ワーカ数に関する Wilcoxon 検定
Table 3 Wilcoxon test of the number of workers

N_T	2	4	6	8	2	4	6	8
CPU	Intel(R) Core(TM) i7				AMD Phenom(TM) II X6			
f_1	*	**	**	**	–	**	**	**
f_2	–	–	**	*	–	–	–	**

表 4 マルチコア CPU の種類に関する Wilcoxon 検定
Table 4 Wilcoxon test of the kinds of multi-core CPUs

N_T	1	2	4	6	8
f_1	–	–	**	**	**
f_2	–	–	**	**	**

〔CDE のサブルーチン（ワーカ W_n）の手順〕

手順 1: 全個体 $\boldsymbol{x}_i \in \mathbf{P}_n$ について $f(\boldsymbol{x}_i)$ を計算する．世代数を $g = 0$ とする．
手順 2: 各ターゲットベクトル $\boldsymbol{x}_i \in \mathbf{P}_n$ について，手順 2.1 から手順 2.3 を実行する．
手順 2.1: ランダムに親個体 $\boldsymbol{x}_{r1}$，$\boldsymbol{x}_{r2}$，$\boldsymbol{x}_{r3} \in \mathbf{P}$（$i \neq r1 \neq r2 \neq r3$）を選択する．
手順 2.2: トライアルベクトル $\boldsymbol{u}$ を生成し，$f(\boldsymbol{u})$ を計算する．
手順 2.3: $f(\boldsymbol{u}) \leq f(\boldsymbol{x}_i)$ なら，$\boldsymbol{x}_i = \boldsymbol{u}$，$f(\boldsymbol{x}_i) = f(\boldsymbol{u})$ とする．
手順 3: $g < G_M$（G_M は世代数の最大値）なら，$g = g + 1$ として手順 2 に戻る．

3. 数値実験と統計的検定

単峰性の Sphere 関数 f_1 と多峰性の Griewank 関数 f_2 を目的関数 $f(\boldsymbol{x})$ とし，$D = 30$ として，それぞれの関数最適化問題に Java 言語で実装した CDE を 30 回ずつ適用した．ただし，制御パラメータは $N_P = 144$，$F = 0.5$，$C_R = 0.9$，$G_M = 1000$ とした．2 種類のマルチコア CPU における CDE の実行時間の平均値を表 1 に示す．括弧内の値は標準偏差である．同様に，CDE で得られた最良解 $\boldsymbol{x}_b$ に対する目的関数値 $f(\boldsymbol{x}_b)$ を表 2 に示す．

ノンパラメトリック検定（Wilcoxon 検定）[4] により，ワーカ数が $N_T = 1$ と $N_T \geq 2$ の場合で目的関数値 $f(\boldsymbol{x}_b)$ を比較した結果を表 3 に示す．記号「*」と「**」は「ワーカ数 N_T の違いで $f(\boldsymbol{x}_b)$ に差はない」とする帰無仮説がそれぞれ危険率 5%と 1%で棄却できることを意味する．同様に，マルチコア CPU の違いで $f(\boldsymbol{x}_b)$ を比較した結果を表 4 に示す．

4. おわりに

CDE は複数のコアを活用して実行時間を短縮するが，CDE により得られる解の精度は，マルチコア CPU の種類のみならず，ワーカ数にも影響されることを示した．今後の課題は，実行時の環境の差異に対してロバストな CDE の実装方法を考案することである．

参考文献

1) C. Bresbears（千住治郎 訳）：並行プログラミング技法，オーム社 (2009).
2) R. Storn and K. Price：Differential evolution - a simple and efficient heuristic for global optimization over continuous space, *Journal of Global Optimization*, Vol.11, No.4, pp.341–359 (1997).
3) K. Tagawa and T. Ishimizu：Concurrent differential evolution based on MapReduce, *International Journal of Computers*, Vol.4, No.4, pp.161–168 (2010).
4) 岩崎学：ノンパラメトリック法，東京図書 (2006).