遺伝的アルゴリズムのための並列処理フレームワークの提案と実装

山中 亮典
Ryosuke YAMANAKA

1 はじめに

大規模な最適化問題を解くために遺伝的アルゴリズム (genetic algorithms: GA) が用いられている．多点探索による繰返し演算を行う GA は並列処理との親和性が高く，様々な並列 GA が提案されている [1]．一方で，マルチコア CPU や GPU など様々な構成のハードウェアが普及している．アーキテクチャ間で実装の互換性が無く，対象ハードウェアを変更するにはプログラムを変更しなければならない．また，並列プログラミングは複雑かつ煩雑であり，生産性に欠ける．

GA 開発者の生産性を向上するために，GA の並列処理フレームワーク GAROP(Genetic Algorithms framework for Running On Parallel environments)[2] を提案する．ユーザ (GA の開発者) は並列処理に関する特別な知識を必要とせず，一般的なプログラミングと同等の記述でマスタ・スレーブ型 [1] の並列化を実現できる．

本研究では Windows クラスタ，マルチコア CPU，GPU に対して GAROP を実現するライブラリを作成する．本稿では，ライブラリを用いて構築した GA を評価し，逐次プログラムと同等の記述で約 3〜6 倍に速度が向上することを示す．

2 GAROP

GAROP の目的は，ユーザが並列化プログラミング技術を有する必要なく，マスタ・スレーブ型の並列処理を実現することである．GA を並列計算環境下で実行する際のプログラミングモデルを定義する．

2.1 要件

GAROP では，以下の要件を満たすべきである．

任意の GA を実行可能な並列化方式 ユーザが用いる計算資源の構成を意識せず，どのような GA でも並列化出来る方式が必要である．

逐次プログラムと同等の生産性 ユーザの負担を軽減するため，どのような並列計算環境であっても共通のプログラム記述で使用できる必要がある．

2.2 ユーザの負担を軽減するフレームワークと設計

ユーザは実行環境を準備した後，任意の GA を構築し評価部以外の部分を実装する．その後，評価関数テンプレートを用い計算資源が実行可能な対象問題のコードを実装する．このフレームワークにより，ユーザは並列処理に関する知識が無くとも，実行する計算環境に適したアルゴリズムを構築できる．

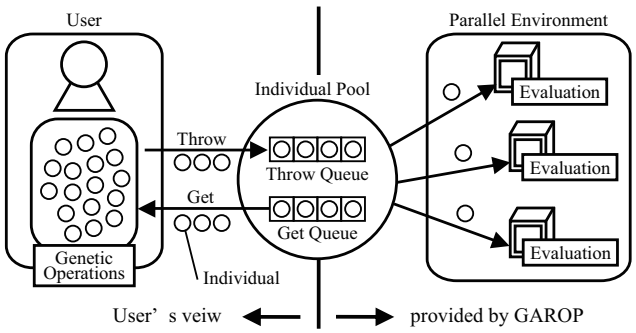

Fig. 1　Concept of the individual pool.

次節より，ユーザと並列計算環境を結ぶインターフェースとして，個体プールの概念を提案する．また，コーディング時に使用する API(application programming interface) について説明し，評価関数のテンプレートについて説明する．

2.2.1 個体プール

Fig. 1 に示す個体プールは個体の溜まり場であり，評価すべき個体及び評価済み個体が格納される．そして，内在する評価すべき個体は自動的に並列評価される．ユーザは評価したい個体を個体プールに登録する．その後，必要な時に個体プールから個体を取得することで評価済みの個体を得ることができる．個体プールにより，どのような GA でも評価計算を並列化できる．

2.3 プログラミングインターフェース

GAROP はライブラリ形式で実現され，基本的に Table 1 に示す 4 つの関数を提供する．並列処理システムの初期化，個体データを個体プールに登録，個体プールから個体データを取得，そして並列処理システムとの接続を切断する関数である．これらの関数を使用することで，逐次プログラムと同等の記述で並列処理を実行する．

2.3.1 評価関数テンプレート

評価計算を実行するのは並列計算環境下の計算資源である．そのため，評価関数は使用する計算機上で実行可能な記述を行う必要がある．対象問題に依存する評価計算の実装を GAROP 提供者が担うのは現実的でないため，評価計算の実装はユーザが行う．この時，GAROP

Table 1　API of GAROP.

Name	Function
initialize	initialize parallel resources.
throw	throw an individual to pool.
get	get an evaluated individual from pool.
finalize	disconnect parallel resources.

List. 1　Source code with the GAROP libraries.

```
1  Individual[] population = InitPopulation();
2  // initialization of GAROP
3  Initialize( sizeof(Individual) );
4  for( j = 0; j < generation_limit; j++ ) {
5    for( i = 0; i < population_size; i++ )
6      // throw individuals to Individual Pool
7      Throw( (BYTE*)&population[i] );
8    for( i = 0; i < population_size; i++ )
9      // get individuals from Individual Pool
10     Get( (BYTE*)&population[i] );
11   selection( population );
12   crossover( population );
13   mutation( population );
14 }
15 Finalize(); // finalization of GAROP
```

の提供するテンプレートを用いることで，特殊な通信と評価タスクのスケジューリングをユーザから隠蔽できる．

3　GAROP の評価

現在作成している GAROP ライブラリは Windows クラスタ，マルチコア CPU，および GPU である．本章では，マルチコア CPU および GPU ライブラリを用いた GA の，記述プログラムおよび速度向上率に関して評価する．GAROP に基づいて実装した SGA(simple GA)[3] を並列実行し，対象環境でのシングルコア実行時と比較する．使用マシンは 6 GB のメモリを搭載し，プロセッサとして Intel Xeon W3530 Quad-core を 2 基搭載した Debian GNU/Linux 5.0.10 のコンピュータである．また，搭載している GPU は Tesla C2050 アーキテクチャで，448 個の CUDA コアを保有する．

対象問題として，大規模問題を想定し演算量を 100000 倍にした 1-max 問題を使用する．個体数は 64 であるため，最大の並列度数は 64 である．

3.1　実験結果：プログラム記述量

List. 1 にマルチコア CPU および GPU 用 SGA の一部を示す．C 言語と CUDA 言語は記述方法がほぼ同一であり，違いは並列処理に関する部分のみである．マルチコア CPU と GPU は全く同様のコードであった．また，両コード共に一般的な関数を用いる記述のみであることが確認できた．

3.2　実験結果：速度向上率

Fig. 2 にシングルコア実行時を 1 とした場合の，GAROP ライブラリを用いた並列実行時の速度向上率を

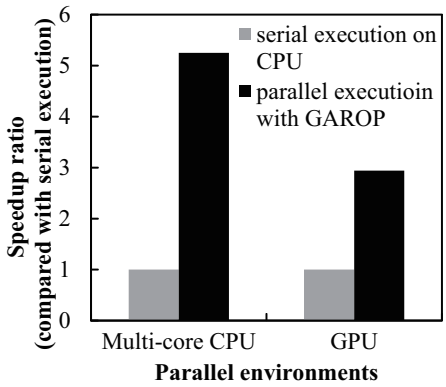

Fig. 2　Speedup on parallel environment compared with executing with single resource.

示す．結果として，マルチコア CPU では 7 スレッド使用で 5.25 倍，GPU では 64 CUDA スレッドで 2.94 倍の速度向上が確認できた．

4　おわりに

GA 開発者の負担を軽減するための，GA の並列処理フレームワーク GAROP を提案し，GAROP を実現するライブラリを作成してその評価を行った．GAROP は，アーキテクチャを問わない将来に渡って連続性のあるアルゴリズムの開発を支援するフレームワークである．ユーザと並列環境とのインターフェースとして個体プールの概念を導入することで，並列処理特有の技術をユーザから隠蔽できる．

マルチコア CPU および GPU ライブラリを用いて構築した SGA とシングルコア実行の SGA を比較し，マルチコア CPU では 5.25 倍，GPU では 2.94 倍の速度向上を確認した．その際の記述コードは評価計算部をライブラリ関数に置き換えるのみであり，逐次プログラムとほぼ同等，すなわち並列処理に関する記述を全く行わなかった．特殊な技術，知識を必要としないコーディングで並列計算環境を使用し，速度向上を実現する GAROP は非常に有用であると考えられる．

参考文献

1) Erick Cantú-Paz. A Survey of Parallel Genetic Algorithms. *Calculateurs Parallels, Reseaux et Systems Repartis*, Vol. 10, No. 2, pp. 141–171, 1998.

2) Tomoyuki Hiroyasu, et al. GAROP: Genetic Algorithm framework for Running On Parallel environments. 数理モデル化と問題解決研究報告, Vol. 2012, No. 5, pp. 1–6, 2012.

3) David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.