並列分散遺伝的アルゴリズムの実装

An Implementation of Parallel Distributed Genetic Algorithm

川崎 高志 (36000714)
Takashi KAWASAKI

Abstract: In this paper, the distributed genetic algorithm library is developed. The developed library is called "GAPPA." GAPPA can be operated not only on sequential computers but also on the parallel computers: shared memory parallel machines and distributed memory parallel machines. Especially, because GAPPA uses the MPI library for message passings, GAPPA can be performed on many types of architectures. In this paper, the concept and the implementation of GAPPA is explained. Comparing GAPPA with some GA Libraries, we achieved successful results.

1　はじめに

遺伝的アルゴリズム (Genetic Algorithms : GA) は生物の進化を模倣した確率的な最適化アルゴリズムである[1)]．この手法は，従来の最適化手法では解くことが困難であった複雑な連続および離散的問題に適用できる上，実装も比較的容易であるという長所がある．しかしながら，GA は膨大な反復計算を必要とするため，計算コストが極めて高いという問題点もある．これまでに多くの GA のためのフレームワークやライブラリが提供されているが，ほとんどが，開発されてからかなりの年月が経過している．したがって，近年, 非常に利用が進んでいる並列処理に対応しているものは少なく, かた, MPI で実装された例は非常に少ない．また, GAPPA は共有メモリ型並列計算機に対応できる．特に分散メモリ型では, MPI によるメッセージパッシングライブラリを使用しているため, 多くの並列アーキテクチャに対応できる．本研究では, GAPPA の概要を述べ, 他の GA ライブラリとの性能評価を行った．

2　遺伝的アルゴリズムの実装

2.1　評価

評価は，すべての個体に対して評価値を求める処理である．したがって，評価関数が複雑である場合には，非常に多くの時間を要する．GAPPA では，既に計算済みの適合度を記憶し，個体の情報が変化するまでは再計算を行わないことで処理速度の向上を図っている．

2.2　選択

選択は，評価された個体の適合度をもとに，次世代の母集団を生成する遺伝的オペレータである．選択については多くの研究がなされており，様々な手法が提案されている．

2.2.1　スケーリング関数による選択

GAPPA では，個体の関数値 e に加えて，さらに全個体の最大値 e_{max} と最小値 e_{min} を採る関数 $F(e, e_{max}, e_{min})$ を使って，$f = F(e, e_{max}, e_{min})$ とし，さらにスケーリング関数が無限の記憶を持つことを許すことにより，他の多くの選択手法をスケーリングで代用することを可能にした．

2.2.2　ランキング選択

ランキング選択は適合度によって各個体に順位付けし，順位に応じた確率で個体を選択するという手法である．この処理は式 (1) で表せる．GAPPA では，この関数とよく似た挙動を示す関数（式 (2)）を用いて，疑似ランキング選択とした．

$$m(g) = \left\lfloor \frac{2n+1-\sqrt{(2n+1)^2-4n(n+1)(1-g)}}{2} \right\rfloor \tag{1}$$

$$m(g) = \lfloor n(1-\sqrt{g}) \rfloor \tag{2}$$

2.3　交叉

交叉においては, ビットパターンの生成部分についての高速化が可能である．特に，m 点交叉におけるビットパターン作成は複雑であり，高速化の余地は大きい．GAPPA の実装では，染色体のビット数分のシャッフル済みの数値リスト (128 ビットならば，0〜127 までのカードをシャッフルしたもの) を用意し，その中から，一つずつ数値を取り出して，該当のビットから染色体の最後までのビットを反転するという方法を採用した．

2.4　突然変異

突然変異は，通常の実装では，すべての個体のすべてのビットに対して適当な乱数を発生させ，その値が突然変異率よりも小さければそのビットを反転させる．した

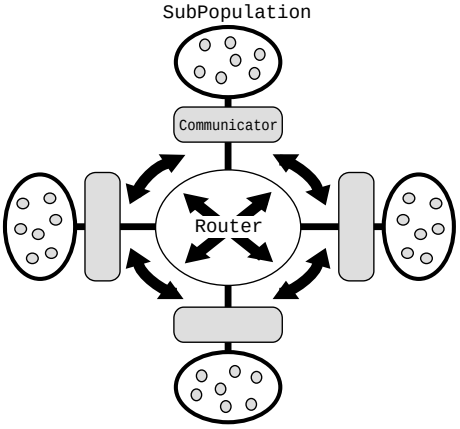

Fig. 1　位置透過性を実現するモデル

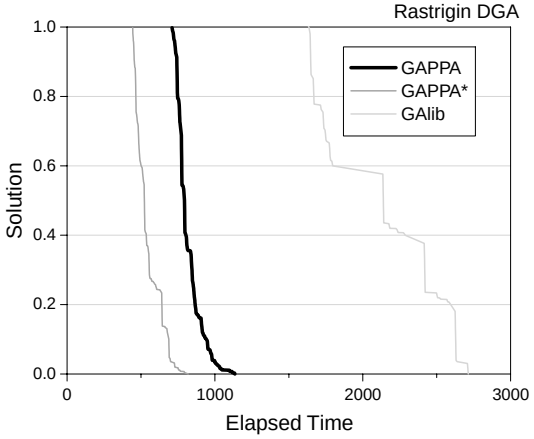

Fig. 2　GAlib との速度比較 (DGA)

がって，染色体長が長ければそれに伴ってこの処理のコストは高くなる．突然変異が n ビット以下である確率 $P_m(n)$ は，式 (3) で求められる．ここで，突然変異の数を決めるために，定義域 [0 1) の乱数 g を発生させるとすれば，式 (4) を満たすような n の値が，突然変異の数である．また，$P_m(n)$ は，染色体長 l，突然変異率が p_m のみに依存するため，この値は，それらのパラメータが設定された時点で確定することが出来る．従って，GAPPA の実装では，突然変異率に変更があった場合にのみこのテーブルを再計算するようにした．実行時には，効率よく n を探し出すことのみが問題となる．

$$P_m(n) = \sum_{k=0}^{n} P_{mutate}(k) = \sum_{k=0}^{n} {}_lC_k p_m^k (1-p_m)^{l-k} \quad (3)$$

$$P_m(n-1) \leqq g < P_m(n) \quad (4)$$

2.5　分散メモリと位置透過性

GAPPA では，共有メモリ型の DGA と MPI クラスタでの PDGA のプログラミングにおけるプログラミングスタイルのギャップを避けるために，コミュニケータという概念を導入し，サブ母集団間の通信は，このコミュニケータを介してのみ行えるようにしている (Fig. 1)．コミュニケータは，移住トポロジを決定するオブジェクトであるルータを使って移住個体の移住先を決定する．コミュニケータは，通信経路を完全に抽象化するため，結果的に位置透過性が実現する．そして，ルータの実装により新しい移住トポロジを設計することが可能になっている．

2.6　他の DGA ライブラリとの比較

他の DGA ライブラリと GAPPA の性能比較を行う．比較対象として GAlib を用いた．対象問題は 10 次元の Rastrigin 関数であり，GA のパラメータは染色体長 100 ビット，母集団サイズ 100，エリート数 10，突然変異率 0.01，交叉率 0.6, サブ母集団数 8，移住率は 0.1 とした．解精度と経過時間の推移を Fig. 2 に示した．

Fig. 2 において GAPPA*としたものは，いくつかのパラメータにチューニングを施したものである．GAlib とノーマルの GAPPA を比較すると, GAPPA は速度の面において非常に良好な結果を示した. また, 高速化のためのオプションを使用することによって，さらなる性能の向上を達成することもできた.

3　結論

PC クラスタへの実装を考慮した並列分散 GA ライブラリ，”GAPPA”の開発を行った．GAPPA と他の代表的な GA ライブラリと比較では，SGA においては速度の面で高い性能を示したが，DGA においては GAlib の方が高速であった．しかしながら，最適なスキームの選択など簡単なチューニングによって，かなりの性能改善が図れることもわかった．高速化のための最適化が今後の課題である．

参考文献

1) D.E.Goldberg, ”Genetic Algorithms in Search Optimization and Machine Learnig”, Addison-Wesley, Reading, Mass.(1989)

2) Reiko Tanese, ”Distributed Genetic Algorithms”, Proc. 3rd International Conference on Genetic Algorithms, P.434-439. (1989)

3) **GENESIS Version 5.0** Copyright (c) 1990 by John J. Grefenstette.

4) **GAlib 2.2.4** Copyright (c) 1994-1996 MIT, 1998-1999 Matthew Wall. `http://lancet.mit.edu/ga/`

5) M.Miki, T.Hiroyasu, M.Kaneko, K.Hatanaka, ”A Parallel Genetic Algorithm with Distributed Environment Scheme”, EEEProceedings of Systems, Man and Cybernetics Conference SMC’99.(1999)