修士論文

ストリーム・プロセッサに適した遺伝的アルゴリズムの提案

同志社大学大学院 工学研究科 情報工学専攻
博士前期課程 2010 年度 764 番

山下　尊也

指導教授　三木 光範教授

2011 年 1 月 18 日

Abstract

This paper proposes a genetic algorithm running on stream processor. We attempt to process the stream flow data as an individual. The individual flows to the stream processor, the individual evolves. Our proposed method is referring to Dual DGA that proposed by Hiroyasu Tomoyuki and Sano Masaki. Therefore, Our method is compared with the Dual DGA. And it is found that our method is inferior to Dual DGA in the ability to search the solution. However, the solution searching ability may be compensated by increasing the number of processing units. Therefore, if the number of processing units increase, our method will become a very profitable.

目 次

1 序論

CPU の急速な発展に従い，クロック数に限界が見えてきた．これらの背景には発熱と消費電力の増大が関係しており，クロック数が増えれば増えるほど CPU は高熱を発するようになる．そのため，CPU 主要メーカである Intel や AMD は 2005 年にシングルコア CPU からマルチコア CPU の開発へとシフトし，現在に至っている．最近ではラップトップコンピュータにもマルチコア CPU である Intel Core i7 などが搭載されている．さらにワークステーションや高性能サーバなどには 6 コアプロセッサを 8 基接続させ，合計 48 コアとした HP ProLiant DL785 G6 などが発売されている．これらの背景から考えると，今後も CPU コア数が増加することは明確である．

しかしながらムーアの法則から CPU コア数を増えれば増えるほど性能向上を見込めるわけではなく，ムーア法則より速度向上率が線形にならないことが導かれている．そのため，現在のプロセッサとは処理の手順を変えたストリーム・プロセッサの研究が昔からなされている．このストリーム・プロセッサを効率よく利用することにより，線形に近い形で速度向上を目指している．特にスタンフォード大学の William Dally 教授が率いる Imagine は 2002 年に 3.5GFPS の性能をコアの消費電力わずか 1.6W で達成している．また，ストリームプロセッシングが利用できるプロセッサとしては，近年数値計算などの分野で注目されている GPU(Graphics Processing Unit) や PlayStation 3 や Cell レグザなどで利用されている Cell Broadband Engine などがあげられる．これらの背景からストリーム・プロセッサが効率的に利用することで，既存のプログラムをより効率よく処理することができると考えられる．

科学計算や様々な分野で利用されている最適化問題は，現在 PC クラスタを利用し計算時間の短縮を狙っている．しかし，ストリーム・プロセッサを利用することで，安価で高速に既存の環境を実現できる可能性がある．そのため本論文では，ストリーム・プロセッサに適した最適化問題を解く手法として，遺伝的アルゴリズムの提案を行う．

本論文の構成を以下に示す．2 章では GA の基本アルゴリズムについて述べ，3 章ではストリーム・プロセッサの構造などについて述べる．そして，4 章では，ストリーム・プロセッサに適した GA の設計について述べ，5 章で評価を行い，6 章で結論を述べる．

2　遺伝的アルゴリズム

2.1　遺伝的アルゴリズムの概要

遺伝的アルゴリズム (Genetic Algorithm:GA) は生物の進化の過程を工学的に摸倣した確率的な最適化手法である [?)?)]．GA ではある世代 (Generation) を形成しているいくつかの個体 (Individual) の集合を母集団 (Population) と呼ぶ．また GA では自然界の進化過程と同様に，環境への適合度 (Fitness) の高い個体が高い確率で選択 (Selection) される．そして，その個体に対して，交叉 (Crossover) や突然変異 (Mutation) がある確率で発生することにより次世代の母集団が形成され，最後に得られた母集団の中で，最も適合度が高い個体を最適解として採用する．

個体は，染色体 (Chromosome) によって特徴付けられており，染色体は複数の遺伝子 (Gene) で構成されている．通常 GA では 1 染色体で 1 個体を表現する．染色体上で各遺伝子の置かれている位置を遺伝子座 (Locus) といい，実数最適化においては対立遺伝子 (Allele) として，Fig. 2.1 のように，2 進数のビット {0,1} を用いることが多い．一方で離散最適化においては，遺伝子型に実数値を適用することが多い．適合度が大きいものほど子孫を残しやすく，小さいものほど死滅しやすいようになっている．このことにより，次世代の各個体の適合度は前世代よりも良いことが期待される．

2.1.1　遺伝的アルゴリズムの基本動作

GA の基本動作のフローチャートを Fig. 2.2 に示す．GA のアルゴリズムの各プロセスの内容は以下のとおりである．

初期化 (Initialization) あらかじめ設定された数だけの個体を生成する．生成される個体の数を母集団サイズ (Population size), または個体数と呼ぶ.

評価 (Evaluation) 各個体に適合度を設定する．適合度は一般的に非負であり，適合度の高い個体ほど良好な個体といえる．

選択 (Selection) 生物の適者生存を模倣したものである．適合度に依存した一定の規則に従い，次世代に残す個体を選択する．選択により，適合度の低い幾つかの個体は淘汰され，その個体数だけ適合度の高い個体が増殖する．代表的な選択の方法として，ルーレット選択 (Roulette selection), トーナメント選択 (Tournament selection) などがある．ルーレット選択は個体の適合度に比例した割合で選択する手法である．トーナメント選択は，集団の中から重複を許して，あらかじめ定められた数の個体をランダムに選び出し，その中で最も適合度の高い個体を選択する操作を，設定された個体数が選ばれるまで繰り返す手法である．この 2 つの選択手法は，適合度の低い個体も選ばれる可能性もある．またこの他に，適合度の高い個体を無条件に次世代に残すエリート保存戦略 (Elitism) も選択手法の 1 つとして考えられる．

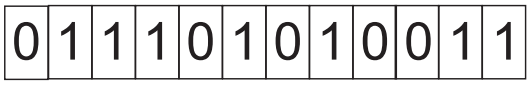

Fig. 2.1　遺伝子型のビット表現

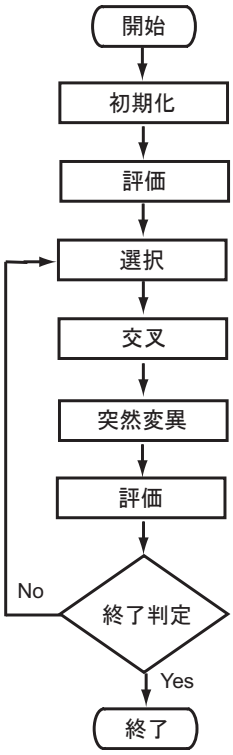

Fig. 2.2　GA のフローチャート

交叉 (Crossover) 生物の有性生殖を模倣したものである．交叉では，親個体の優れた部分形質を子個体に継承することを目的としている．交叉の仕組みは，定められた交叉率 (Crossover rate) に基づき，個体の遺伝子と別の個体の遺伝子を入れ替えることにより，新しい子個体を生成するものである．優れた個体同士が交叉すると，それぞれの個体の優れた部分解を構成する部分が結合し，より優れた個体が生成されることが期待される．

突然変異 (Mutation) 選択，交叉のみでは，初期母集団内の遺伝子に依存するような限られた範囲の子個体しか生じない．そのため，一般にあまり望ましくない解に収束することが多い．従って，突然変異によって選択・交叉だけでは生成できない子個体を生成し，個体群の多様性を維持することが必要である．

突然変異では，染色体上のある遺伝子座を，他の対立遺伝子に置き換える．突然変異は突然変異率 (Mutation rate) で定められた確率で起こる．これは自然界における DNA 複写の際のコピーミスに当たる．

終了判定 (Terminal criterion) あらかじめ定められた終了条件に基づいて，GA の処理を終了する．この時の母集団で適合度の最も高い個体を最適解として採用する．

2.2　分散遺伝的アルゴリズム

分散遺伝的アルゴリズム (Distributed Genetic Algorithm:DGA) は GA と同様に生物の遺伝と進化を模擬した多点探索手法である [?]．DGA では，GA における母集団を複数のサブ母集団に分割し，

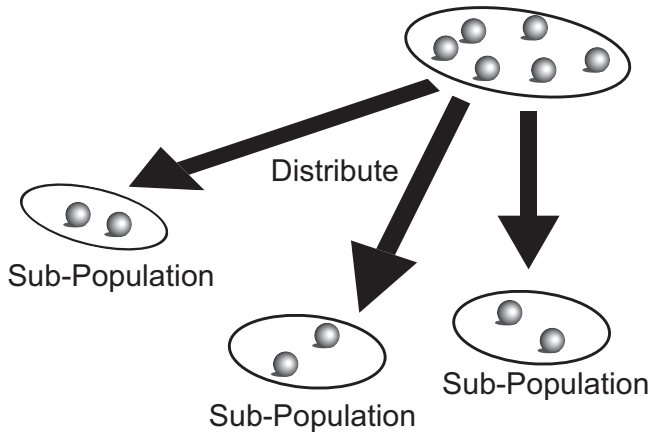

Fig. 2.3　DGA の概念図

各サブ母集団 (Sub-Population) ごとに独立に遺伝的操作を行い，一定世代ごとに異なるサブ母集団間で移住と呼ばれる複数個体の交換を行う．結果として，全ての個体が一つの母集団を形成するよりも多様性が大きくなり，より効率的な探索を進めることが可能である．ここで，移住 (Migration) を行う世代間隔を移住間隔 (Migration interval) と呼び，サブ母集団の個体数に対する移住個体の割合を移住率 (Migration rate) と呼ぶ．DGA と移住の概念を Fig. 2.3 に示す．

DGA の特徴として，母集団の分割による利点がある．母集団を分割することにより各島で個体の異なった遺伝子が進化している場合，移住によって他の島の個体と混ざり合い交叉を行うことで，優れた遺伝子同士が集まった，非常に優れた個体が誕生する可能性がある．

2.2.1　分散遺伝的アルゴリズムの基本動作

GA との違いは，“選択” の後に “移住間隔” であれば “移住” 操作を行うことである．移住は，数世代に一度，各サブ母集団内で選ばれた 1 つまたは複数個の個体 (移住個体:Migrant) を別のサブ母集団と交換することで実現される．移住には，移住元と移住先を結んだ線が 1 つのリングになるように移住先を定める $Randomring$[?)] 手法や，各サブ母集団に ID を付与し，その上で各サブ母集団の ID が隣り合っている 2 つのサブ母集団と移住を行う $bi-DirectionalRing$[?)] など様々な手法が存在する．

2.3　2 個体分散遺伝的アルゴリズム

2 個体分散遺伝的アルゴリズム [?)](Dual DGA) は DGA において 1 島あたりの個体数を 2 としたものであるが，DGA とは異なる遺伝的操作を採用している．Dual DGA の手順を以下に示す (Fig. 2.5). まず，個体の母集団をランダムに，2 個体ずつの島に分割する．各島において次の操作を世代ごとに繰り返す．

交叉 親個体である 2 つの個体 (**A**,**B**) を交叉させ，新しい 2 つの個体 (**A'**,**B'**) を生成する．

突然変異 交叉させ生成した個体 (**A'**,**B'**) を突然変異させ，新しい 2 つの子個体 (**A"**,**B"**) を生成する．この際，個体がもつビット列に突然変異率に従いビット反転させる．

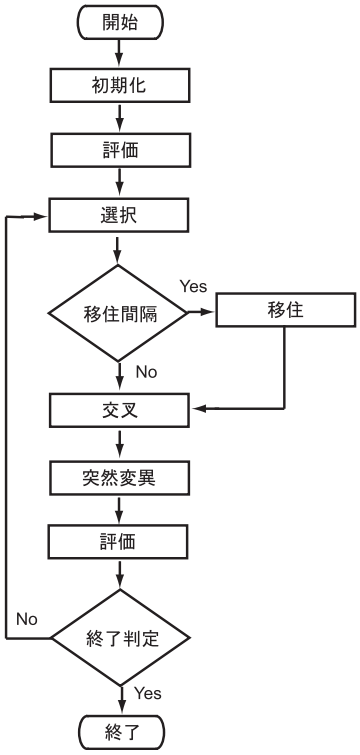

Fig. 2.4　DGA のフローチャート

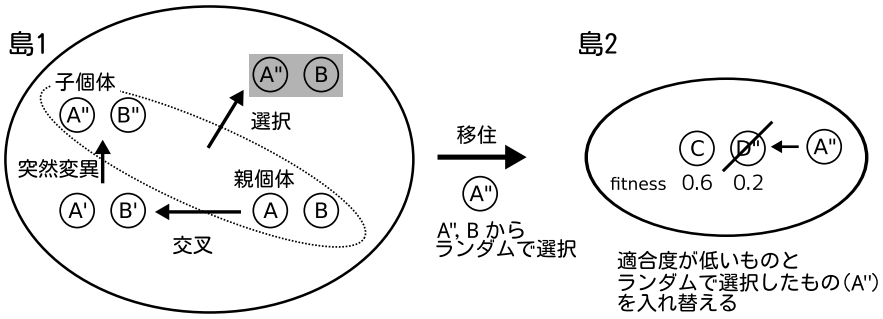

Fig. 2.5　Daul DGA における遺伝的操作

選択 2 つの親個体 (**A**,**B**) と 2 つの子個体 (**A”**,**B”**) から，それぞれ適合度の高い方の個体をび，次世代の親個体とする．

移住 2 つの個体のうち，ランダムに一方を選択し (**A”**), そのコピーを他の島に送る．そして適合度の低い方の個体 (**D”**) は，他の島から送られてきた個体に置き換えられる．移住トポロジは，移住のたびにすべての島が無造作な順番の 1 つのリングを形成し，隣の島が移住先となるものである．また，移住方向は一方向のみである．

3　ストリーム・プロセッサ

3.1　ストリーム・プロセッサ

Fig. 3.1 に示すように，ストリーミングプロセッシングの計算モデルとは，ストリームと呼ばれる入力データセットの各要素に対し，カーネルと呼ばれる演算操作を作用するものである．カーネルはストリームに対してパイプライン化された形で動作し，入力および演算後の出力ストリームは外部のメインメモリでなくプロセッサ内のローカルストアに格納している．

このような構造を持つプロセッサをストリーム・プロセッサと呼ぶ．しかしながら，このような構造を持つストリーム・プロセッサは非常に少ない．代表的なものとして，スタンフォード大学でアーキテクチャ，ソフトウェアツール， VLSI の実装および開発ボードなどを開発している Imagine. そして，同じくスタンフォード大学でストリーム型のスーパーコンピュータの開発を目的とした Merrimac などがあげられる．近年ではソニーコンピュータエンターテインメント，東芝， IBM が開発した Cell Broadband Engine が PlayStation 3 や液晶テレビなどに搭載されている．また，パーソナルコンピュータに搭載される GPU (Graphics Processing Unit) は，ストリーム・プロセッサと見ることができる．AMD と NVIDIA と S3 が，ストリームプロセッシング用途に活用可能な高性能な GPU を製造している．この章では，最も身近に用いられているストリーム・プロセッサとして， GPU について述べる．

3.2　Graphics Processing Unit

GPU (Graphics Processing Unit) は描画処理に特化した専用プロセッサであり，パソコンの拡張スロットに装着するビデオカード (グラフィックス・ボード) や，CPU に付随するチップセットの中に汲み込まれている．

高速に画像を表示することはパソコンやワークステーションにとって非常に重要なことである．これらの処理を CPU に処理させるよりも専用のハードに処理させようという発想から GPU は生まれた．そのため GPU は大規模な並列処理を得意とし，物理シミュレーションなどの科学計算にも向いている．

これまで，いくつもの GPU ベンダーが生まれては消えてきた．現在では NVIDIA 社と AMD 社が高性能 GPU で熾烈な性能競争をしている．

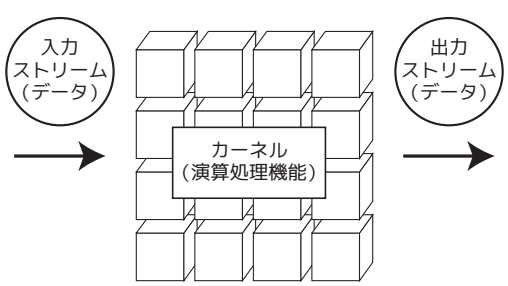

Fig. 3.1　ストリームプロセッシングのイメージ

その性能競争の過程で，次第に高度な描画機能が求められるようになり，プログラマブル・シェーダと言われる技術が登場した．プログラマブル・シェーダとは， GPU 上での陰影処理 (シェーダ) がプログラミング可能な技術のことである．この技術をきっかけに， GPU をグラフィックス処理だけでなく汎用計算に適用できるのではないかと感じる人が現われ始めた．

3.3 GPGPU

GPGPU (General Purpose GPU) は，GPU を画像処理ではなく汎用計算の分野で利用することである．それを象徴するように，グラフィックス・ボードに搭載されている GPU とほぼ同じ性能を持つ GPU を搭載しながら，ビデオ出力端子が付いていないグラフィックス・ボード，いわゆる GPGPU ボードが NVIDIA からは Tesla, AMD からは FireStream (ファイアストリーム) というブランドで発売されている．

GPGPU が最初に盛んに利用され出したのは天体計算の分野であった．この分野では，重力多体問題を計算することで銀河形成の仕組みの解明が行われている．

N 個の星からなる銀河を計算する場合，$N \times N$ 回の重力の相互作用を計算しなければならず，GRAPE と呼ばれる専用計算機 (加速ボード) などが使われてきた．加速ボードを GPU に置き換えることで， GPU が天体計算で高性能を引き出すことを示した．現在ではアストロ GPU という学会があるほどに天体計算の分野では GPU が普及している．

以上のことが引き金となり， GPU は様々な分野で使われ始めた．NVIDIA 社が提供している Web ページ CUDA ZONE では，様々な分野で GPU が使われていることがよく分かる．

GPGPU は演算性能，コスト，設置の容易，この観点から注目されている．以下に，それぞれについて説明する．

演算性能 最近の CPU である Core i7 の理論性能が約 100GFLOPS であるのに対し，NVIDIA 社の GeForce GTX 285 は 1TFlops (1000GFlops) を超える理論性能を持つ．AMD 社のハイエンド GPU も同等である．

スーパーコンピュータ (以下:スパコン) がはじめて 1TFlops を超えた 1997 年から約 10 年で，当時の世界最速スパコンと同等の性能をパソコン 1 台で実現できるようになった．

コスト グラフィックス機能はすべてのパソコンに必要である．ゲームや CG (コンピュータ・グラフィックス) の市場は大変大きくなったため，販売される GPU の個数が多く，コストも低くなっている．

設置の容易 スパコンは非常に特殊な環境でしか設置できないことに対し，GPU は非常にコンパクトであるため，自家用パソコンでも簡単に装着が可能である．また，必要な電力も家庭で十分に補える程度である．

3.4 NVIDIA 社製 GPU のアーキテクチャ

NDIVIA 社製 GPU のアーキテクチャを Fig. 3.2 に示す．GPU の構造は世代や型番で異なるが，基本的なアーキテクチャはほぼ同じである．

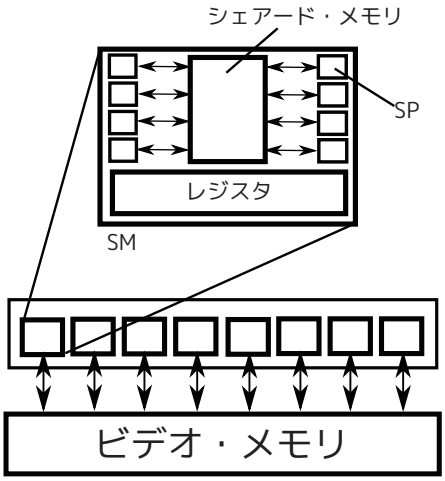

Fig. 3.2　GPU アーキテクチャ

ビデオ・メモリと GPU はメモリ・インターフェイスで接続されている．CPU の場合はメイン・メモリとせいぜい 64bit 幅で接続されているが，ハイエンドな GPU の場合では 512bit のものもあり，ビデオ・メモリと高速にデータ転送ができるアーキテクチャになってる．

GPU の内部にはストリーミング・マルチプロセッサ (SM) が多数入っている．さらに SM の内部にはストリーミング・プロセッサ (SP) と呼ばれる最小単位の演算処理ユニットが 8 個入っている．

両ボードともに GT200 という GPU を採用しており， SM が 30 個搭載されている．従って，1 チップ当たり 240 個の SP が入っていることになる．これは同年代に発売されている Intel 社の CPUCore i7 などが 4 コアであることと比較しても，非常に多いことが分かる．GT200 が持つ SP の動作クロックは 1.48GHz であるため，各 SP が浮動小数点実数の積和演算命令 ($D = A \times B + C$) を 1 クロック当り 1 命令実行すれば $1.48 \times 240 \times 2 = 710.4$GFlops となる．さらに，積和演算命令と乗算命令 ($C = A \times B$) を同時実行することで理論的には 1.06TFlops となり，単精度計算であるが演算性能が 1TFlops を超えることが可能である．

GPU の性能が高い点としてビデオ・メモリのメモリバンド幅の太さが挙げられる．多くの計算はメモリバンド幅に性能が左右されるため，CPU の Core i7 ではメイン・メモリに DDR3 を採用し，さらにメモリ・コントローラをチップ内蔵にすることで 36GB/sec まで太くした．一方で GPU の GeForce GTX 285 ではメモリ・クロックが 2.48GHz, メモリ・インターフェイスが 512bit (64Byte) であるため， 159GB/sec のメモリバンド幅となる．

2009 年 4 月から稼働している地球シミュレータ 2 に採用されている SX-9 の 1 プロセッサ当たりの演算性能が 102GFlops, メモリバンド幅が 256GB/sec となっている．これと比較しても， GPU がスパコンに匹敵する性能を持っていることが分かる．

4　ストリーム・プロセッサに適した遺伝的アルゴリズムの設計

4.1　ストリーム・プロセッサに適した遺伝的アルゴリズム

今回，ストリーム・プロセッサに適した遺伝的アルゴリズムとして 2 つの手法を考案する．1 つは，2.3 で述べた 2 個体分散遺伝的アルゴリズムの移住操作に注目した手法．そして，その手法をさらに改良しリング構造を持った遺伝的アルゴリズムである．

4.2　パイプライン化された Dual DGA

3 章で述べたように，ストリーム・プロセッサに適したアルゴリズムにするためには，データを 1 方向に流す必要がある．すなわち，データがパイプライン化されており，各プロセスにおいてデータの依存関係を明らかにする必要がある．そのため，2.3 を参考に島の中での遺伝子の依存関係に注目し，データを 1 方向に流れるようにした遺伝的アルゴリズムの概念図を Fig. 4.1 に示す．遺伝的操作については 2.3 で述べた 2 個体分散遺伝的アルゴリズムと等しい．

入力 あらかじめ設定された数だけの個体を生成し，最初の島に個体を入力する．その際，島の中で個体が 2 個以上ある場合は島で交叉，突然変異，選択，移住を行い， 2 個未満の場合は遺伝的操作を行わない．

交叉 2 個体分散遺伝的アルゴリズムと同じ手法を用いる．

突然変異 2 個体分散遺伝的アルゴリズムと同じ手法を用いる．

選択 2 個体分散遺伝的アルゴリズムと同じ手法を用いる．

移住 移住間隔に従って移住させる．

出力 最終的の島の移住先を出力先の母集団とする．

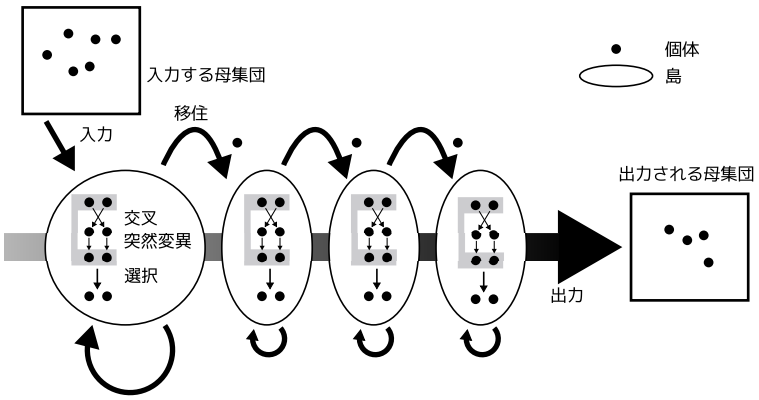

Fig. 4.1　パイプライン化された Dual DGA

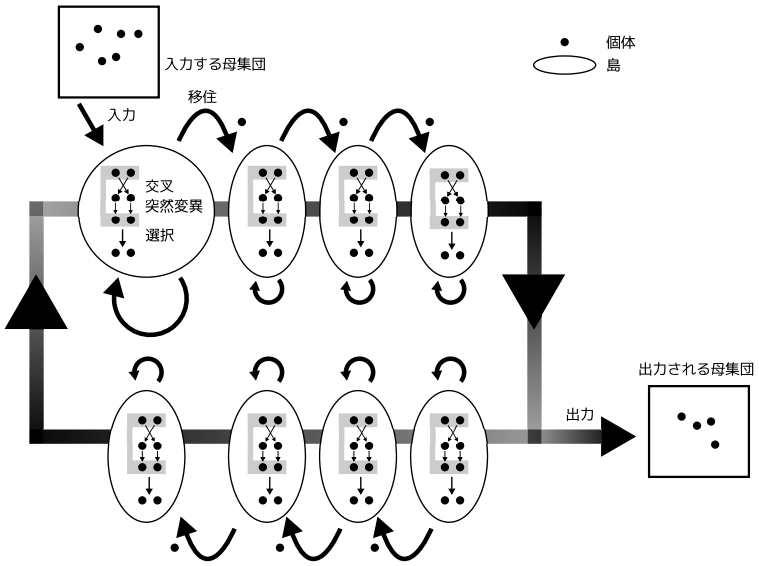

Fig. 4.2　パイプライン化されリング構造を持ったストリーム型 Dual DGA

4.3　パイプライン化されリング構造を持ったストリーム型 Dual DGA

4.2 節のパイプライン化された Dual DGA の個体を永続的に移住させるようにしたリング構造を持った Dual DGA を提案する．パイプライン化されリング構造を持ったストリーム型 Dual DGA の概念図を Fig. 4.2 に示す.

入力 あらかじめ設定された数だけの個体を生成する.

交叉 2 個体分散遺伝的アルゴリズムと同じ手法を用いる.

突然変異 2 個体分散遺伝的アルゴリズムと同じ手法を用いる.

選択 2 個体分散遺伝的アルゴリズムと同じ手法を用いる.

移住 移住間隔に従って移住させる．2 つの個体のうち，ランダムに一方を選択し，そのコピーを他の島に送る．そして適合度の低い方の個体は，他の島から送られてきた個体に置き換えられる．移住トポロジは，すべての島を 1 つに繋げたリングを生成する．そのリング情報をもとに，隣の島が移住先とする.

出力 最終的の島の移住先を出力先の母集団とする.

5 評価

5.1 データがパイプライン化されリング構造を持った遺伝的アルゴリズムの評価

通常の Dual DGA とストリーム型 Dual DGA との性能評価を行った.

本論文で対象としたテスト関数は，Rastrigin 関数， Griewank 関数， Schwefel 関数， Ridge 関数の 4 つである．それぞれの関数を式 (5.1)〜 式 (5.4) に示す．いずれも大域的最適解は 0 である．Table 5.1 に示すパラメータを用いた．特に断りがない限り，以下の実験結果はすべて 20 試行の中央値である.

$$
\begin{aligned}
F_{Rastrigin}(x) &= 10n + \sum_{i=1}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right) && (5.1)\\
&\quad (-5.12 \le x_i < 5.12)\\
min(F_{Rastrigin}(x)) &= F(0,0,\ldots,0)\\
&= 0\\
F_{Griewank}(x) &= 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\left(\cos(\frac{x_i}{\sqrt{i}})\right) && (5.2)\\
&\quad (-512 \le x_i < 512)\\
min(F_{Griewank}(x)) &= F(0,0,\ldots,0)\\
&= 0\\
F_{Schwefel}(x) &= \sum_{i=1}^{n} -x_i \sin\left(\sqrt{|x_i|}\right) - C && (5.3)\\
&\quad (C \text{ is the optimum.})\\
&\quad (-512 \le x_i < 512)\\
min(F_{Schwefel}(x)) &= F(420.968750,\ldots,420.968750)\\
&= -418.98288727 \cdot n\\
F_{Ridge}(x) &= \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2 && (5.4)\\
&\quad (-64 \le x_i < 64)\\
min(F_{Ridge}(x)) &= F(0,0,\ldots,0)\\
&= 0
\end{aligned}
$$

Table 5.1 Dual DGA で用いたパラメータ

Number of individuals	512
Number of elites	1
Number of design variables	30
Chromosome length	Number of design variables × 20
Selection	Roulette selection
Crossover rate	1.0
Crossover	1pt. crossover
Mutation	1 / Chromosome length
Migration gap	5 generations

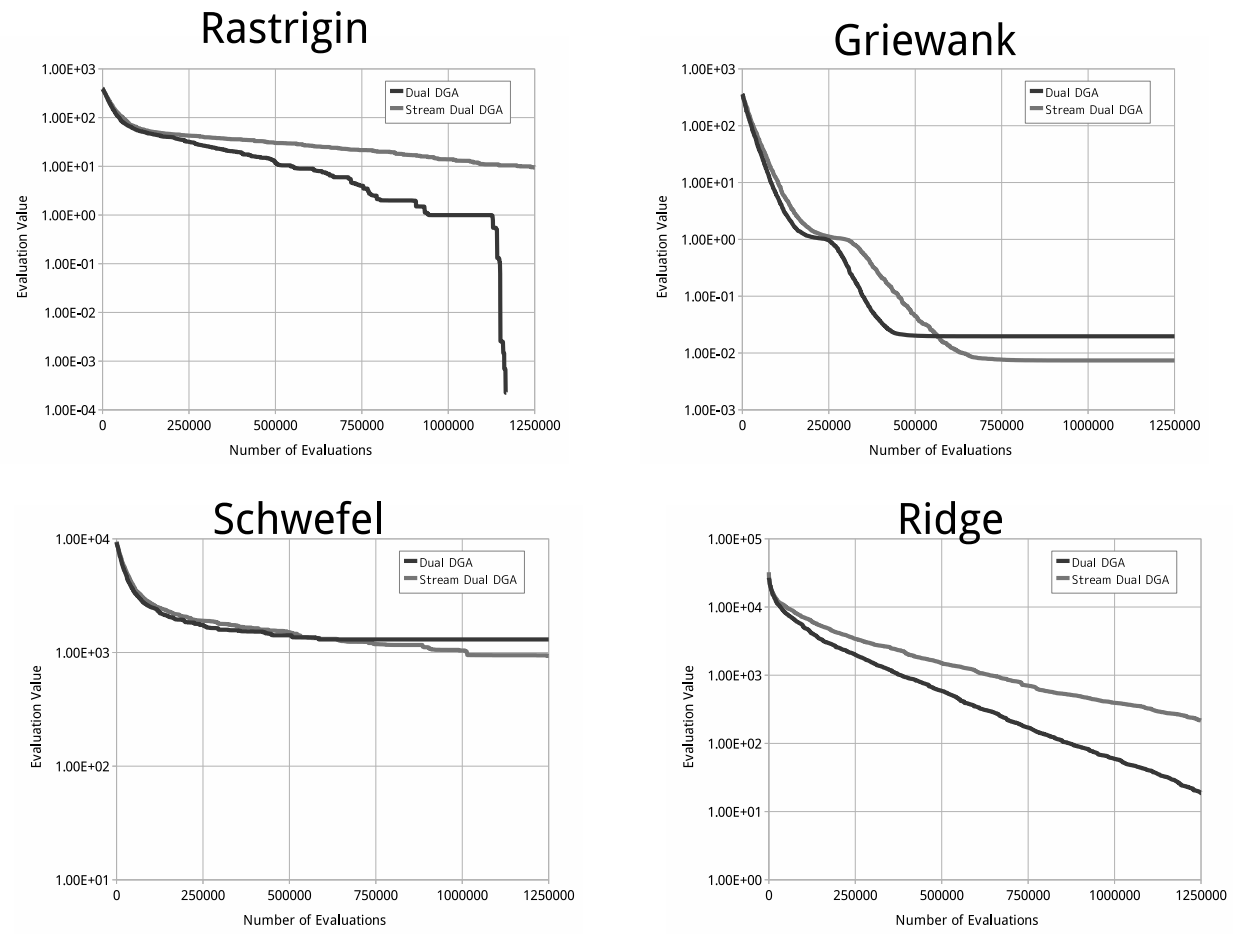

Fig. 5.1　Rastrigin, Griewank, Schwefel, Ridge 関数の評価値履歴

Fig. 5.1 は，それぞれ Rastrigin 関数， Griewank 関数， Schwefel 関数， Ridge 関数を対象問題とし，横軸に関数評価回数をとり，関数評価値の履歴を示したものである．5000 世代での最良個体の関数評価値を Table 5.2 に示す．関数評価値の履歴の平均値を Fig. 5.2 に示す．

解の探索能力の優劣は問題によって異なっている． Rastrigin 関数では，通常の Dual DGA の方がストリーム型 Dual DGA と比較して速く良い解を得ているといえる．また，通常の Dual DGA では大域的最適解に到達しているのに対し，ストリーム型 Dual DGA では大域的最適解に到達していない．

Griewank 関数では，ストリーム型 Dual DGA が通常の Dual DGA よりも速く良い解を得ているといえる．また，通常の Dual DGA でもストリーム型 Dual DGA でも大域的最適解に到達している．

Schwefel 関数では，ストリーム型 Dual DGA が通常の Dual DGA よりも速く良い解を得ているといえる．

Ridge 関数では，通常の Dual DGA がストリーム型 Dual DGA よりも速く良い解を得ているといえる．

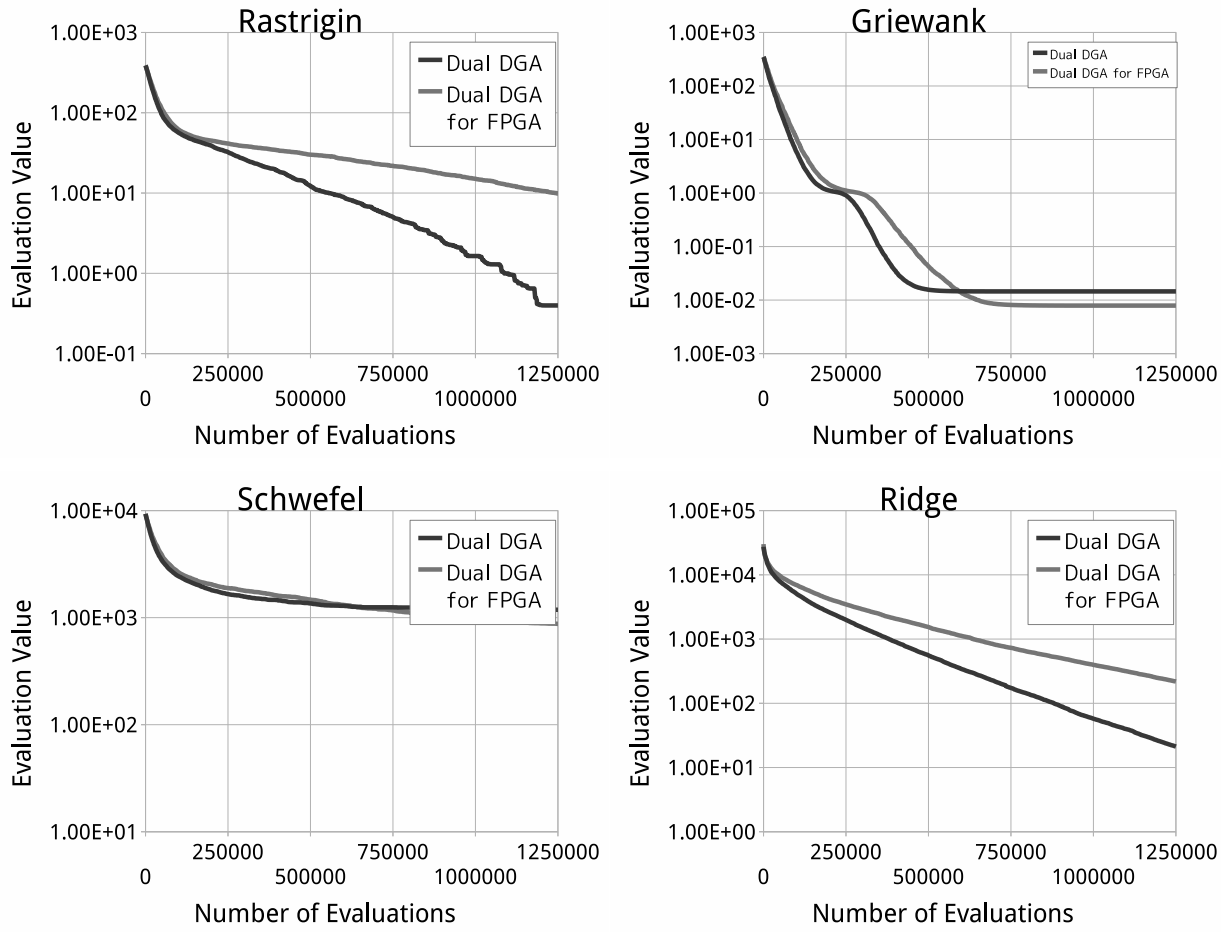

Fig. 5.2　評価値履歴の平均値

Table 5.2　最良個体の関数評価値

	Dual DGA	Dual DGA for FPGA
Rastrigin	0.00000	3.02079
Griewank	0.00000	0.00000
Schwefel	710.62631	229.50622
Ridge	3.72044	79.78689

5.2 ストリーム型 DGA とストリーム型 Dual DGA の比較

5.2.1 実験概要

ストリーム型に変更した DGA と Dual DGA との性能評価を行った．対象としたテスト関数は，Rastrigin 関数である．また，用いたパラメータを Table 5.3 に示す．

5.2.2 実験結果

島数 10 ， 20 ， 30 ， 40 ， 50 ， 60 の場合の DGA と Dual DGA を用いた際の評価回数を Table 5.4 に示す．ストリーム型 DGA ，ストリーム型 Dual DGA での評価回数を基準とした評価値履歴を Fig. 5.3 に示す．

Table 5.4 ストリーム型 DGA とはストリーム型 Dual DGA は島数が増えるにつれ，評価回数が増加している．それに伴いストリーム型 DGA は，島数 40 ，島数 50 ，島数 60 では最適解を見つけている．しかし，島数 30 より少ない場合は評価回数が十分に得られておらず，最適解を見つけることができていない．

しかし，ストリーム型 Dual DGA はストリーム型 DGA に比べ評価回数は多いが，収束の速度が

Table 5.3 DGA ， Dual DGA で用いたパラメータ

Number of input individuals	2000
Number of islands	10,20,30,40,50,60
Number of elites	1
Number of design variables	30
Chromosome length	Number of design variables × 10
Selection	Roulette selection
Crossover rate	1.0
Crossover	1pt. crossover
Mutation	1 / Chromosome length
Migration gap	5 generations

Table 5.4 評価回数

手法	評価回数
DGA 島 10	973200
DGA 島 20	1896200
DGA 島 30	2769200
DGA 島 40	3592200
DGA 島 50	4365200
DGA 島 60	4926200
Dual DGA 島 10	999050
Dual DGA 島 20	1998050
Dual DGA 島 30	2997050
Dual DGA 島 40	3996050
Dual DGA 島 50	4995050
Dual DGA 島 60	5994050

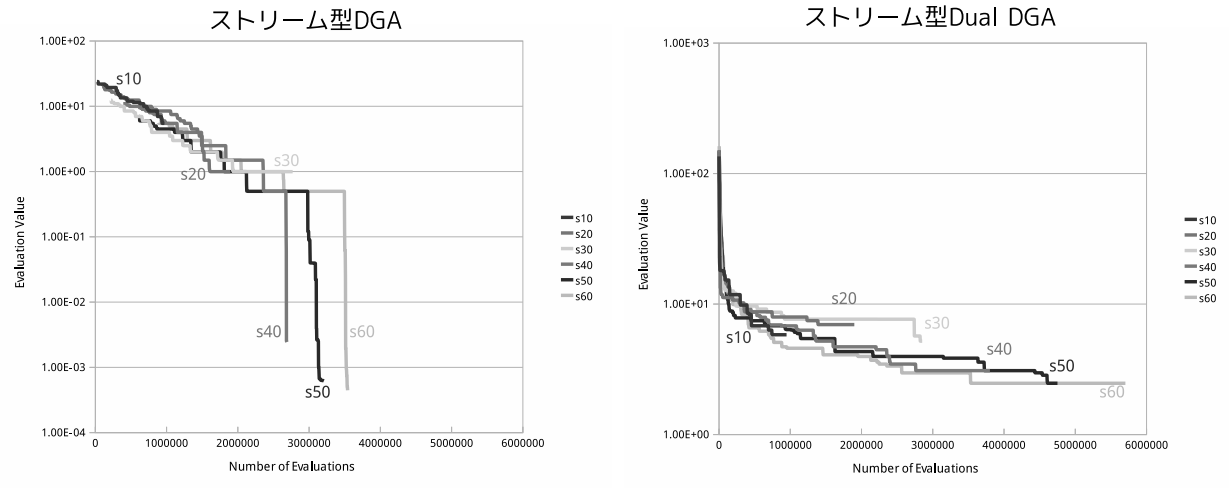

Fig. 5.3　ストリーム型 DGA，ストリーム型 Dual DGA での評価回数を基準とした評価値履歴

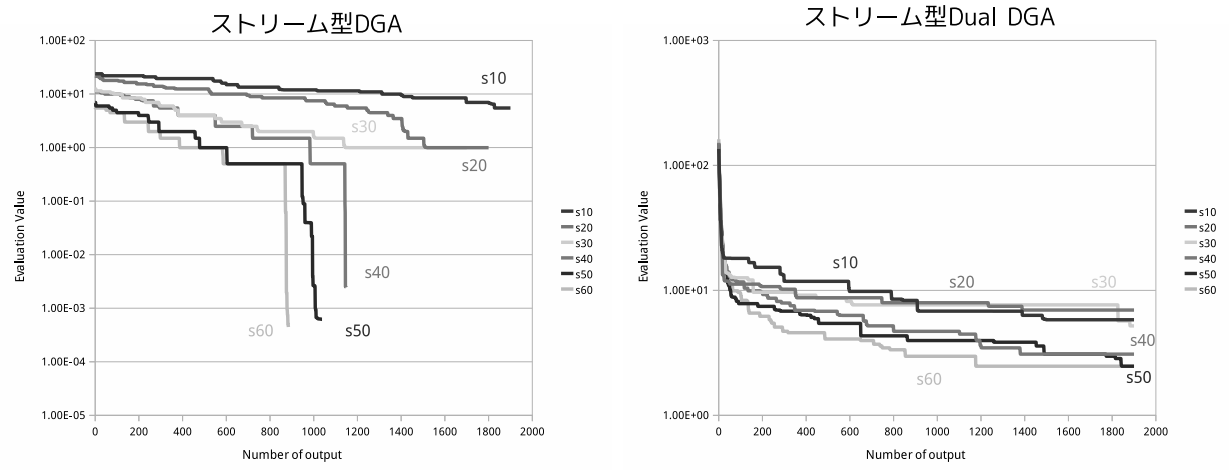

Fig. 5.4　ストリーム型 DGA，ストリーム型 Dual DGA での出力個体を基準とした評価値履歴

ストリーム型 DGA より遅いことが分かる．このことより，ストリーム型 DGA に比べ，ストリーム型 Dual DGA は探索性能が低いことが分かる．

6　結論

本論文では，ストリーム・プロセッサに適した遺伝的アルゴリズムを提案した．その結果，ストリーム・プロセッサを利用した場合，評価計算回数を増やすことができるため，島数を増やせば増やすほど，より短時間で最適解を見つけることができることが確認できた．

また，ストリーム型 DGA とストリーム型 Dual DGA を比較したところ，ストリーム型 Dual DGA の方がストリーム型 DGA に比べ評価回数は多くなるが，ストリーム型 DGA の方が性能が良いことが分かった．

以上より，ストリーム・プロセッサに適した遺伝的アルゴリズムを実装する際は，ストリーム型 Dual DGA では解の多様性が維持されないため，ストリーム型 DGA を用いた方が良いと考えられる．

謝辞

本研究を遂行するにあたり，多大なる御指導そして御協力を頂きました，同志社大学生命医科学部の廣安知之教授に心より感謝いたします．廣安教授は勝手な行動をとった私にも関わらず最後まで御指導して頂きました．

また，様々な指摘，助言をして下さいました，同志社大学理工学部の三木光範教授同志社大学理工学部の吉見真聡助教に心より感謝いたします．吉見助教は研究だけではなく，私の進路についても指導して頂きました．また，サーバやアーキテクチャに関することなどで良く議論などをさせて頂きました．

私の誤字脱字などを指摘していただいた横田山都さん，米田有佑さん，本当にありがとうございました．

付 図

付 表