拡張性を考慮したＳＩＭＤ型並列遺伝アルゴリズムと専用プロセサ要素の設計

4Ｆ－4

井上　富夫†　稲富　裕介*　佐野　雅彦†　高橋　義造†
(†徳島大学工学部　*NEC ICマイコンシステムズ)

１．まえがき

1960年代に開発された遺伝アルゴリズム[1, 2]は，計算機の性能向上によりいくつかの定型的な問題についての効果が確認されている．しかし実際の問題に適用するには，膨大な計算量となり[3]，通常の計算機で行うのは非現実的である．また，従来の逐次型プログラムをそのまま並列化するだけでは，せいぜい数十倍程度の時間短縮にしかならない[4, 5, 6]．本研究の目的は遺伝アルゴリズム専用のマシンを作成することにより，演算速度を画期的に高めることにある．その要求仕様として，ハードウェア化を行うので，ハード的な拡張性が高いこと，つまり個体数を多くとれる必要がある．

これまでにSIMD型遺伝アルゴリズムを提案し，このアルゴリズムに専用的に用いる並列計算機を開発してきた．[9] 本報告では拡張性をもたせるために，SIMD型遺伝アルゴリズム専用プロセサ要素をFarming Modelによる階層化のシミュレーションを行い，システムの方式について検討する．２節ではSIMD型遺伝アルゴリズムについて，３節でプロセサ要素およびシステム構成と拡張性，４節では探索問題への適用，５節はシミュレーション結果ついて述べ，６節にまとめを述べる．

２．SIMD型遺伝アルゴリズム

遺伝アルゴリズムは図１（ａ）に示すように多数の候補解の集合に遺伝子演算を加えて最終的に最適解を求めるものである．このアルゴリズムの問題点は長時間の計算を行う必要があることで，これを短縮するために個体数を制限すると局所解に陥入り，解の信頼性が損なわれる．Grobal Population Model (GPM)は解の集合を共有メモリに置き，これを並列計算機により高速化を目指すが，共有メモリへのアクセスがシリアルでしかアクセスできないため，演算速度には上限が生じる．そこで個体集合を多数の部分集合に分割し，部分集合ごとに交配，淘汰を繰り返し，一定の周期で部分集合間で優秀な解を交換しながら計算を進める分散型遺伝アルゴリズム[7, 8]が提案されているが，この方法ではよい遺伝子が集合全体に伝搬するのが遅れ，収束が遅くなる．SIMD型遺伝アルゴリズムは，個体の集合を分割せず，一つの集合の中で遺伝子演算と淘汰を高速に行うような新しい並列遺伝アルゴリズムである．SIMD型遺伝アルゴリズムを図１(b)に示す．[9] このアルゴリズムでは，各個体は自律的に動作して一斉に交配，突然変異，淘汰を行うものである．すなわち各個体は自己評価を行い，相手の評価の値にもとづき優れた個体をランダムに選び，交差演算を行い子孫を生成する．突然変異はランダムに選ばれた個体自身が変化する．評価値により優秀な相手を選ぶので，優秀な個体が遺伝されていく．

次にこのSIMD型遺伝アルゴリズムの処理手順を示す．

```
SIMD_GA {
    初期個体の発生；
    自己評価；
    目標値が見つかるまでループ {
        良い相手を見つける(selection)；
        if(＃)子供を生む(crossover & mutation)；
        自己評価( evaluation )；
        世代交代；
    } ／＊　ここで　＃は交差率，突然変異率　＊／
}
```

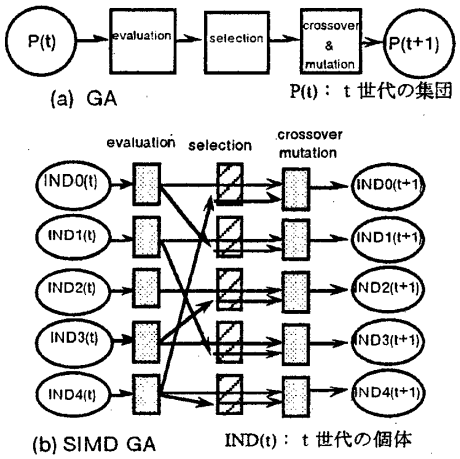

図１．遺伝アルゴリズム (a) とSIMD型遺伝アルゴリズム (b)

３．プロセサ要素，システムの構成，拡張性

SIMD型遺伝アルゴリズム専用プロセサ要素(PE)の構成を図２に示す．PEのバスは３２ビットで構成し，全ての演算が３２ビットで行うことができる．また，共有メモリマトリクス(SMM)は８台のPEがメモリを共有し，それぞれのPEが同時に書き込み，読み出しを行えることにより，プロセサをSIMDで動作できるという特徴がある．プロセサの命令セットは参考論文［9］のものを用いる．

システムの構成を図３に示す．このシステムを用いてFarming Modelによる遺伝アルゴリズムを実行する．このモデルは図４に示すように，初期個体として畑に種を蒔き，交配，突然変異により，次の新しい種を収穫する．収穫された種の中で優良な種だけを農場に集める．農場では優良な種だけで，畑では農場からの種を加えて次の世代の進化を行う．この操作を繰り返すことにより全体の進化を図る．この結果として，優良な種どうしの交配が並列に行われるので，探索範囲が広くなる．

４．探索問題への適用

上記のSIMD型GA専用プロセサを探索問題としてナップザック問題に適用する．item数は３２とし，３２ビットの遺伝子をもつ個体の探索を行なう．itemの最大価値，重量ともに１６とする．SMMのメモリセルの１個体の容量を表１に示す．SMMは８×８個のセルで構成するのでプロセサ８台当たり約６Kbitとなる．評価値はtotal

---

Scalable SIMD Genetic Algorithm and the Design of a Processing Element Adapted to the Algorithm
†Tomio Inoue, *Yusuke Inatomi, †Masahiko Sano, †Yoshizo Takahasi
†Faculty of Engineering, University of Tokushima.
*NEC IC Microcomputer Systems.LTD

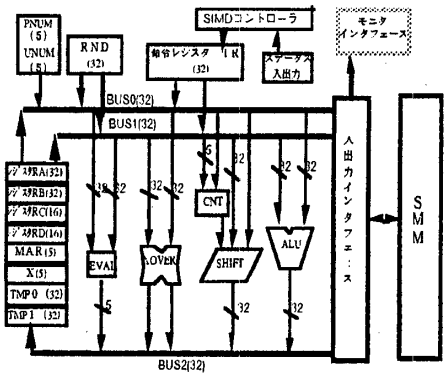

図２．プロセサのアーキテクチャ

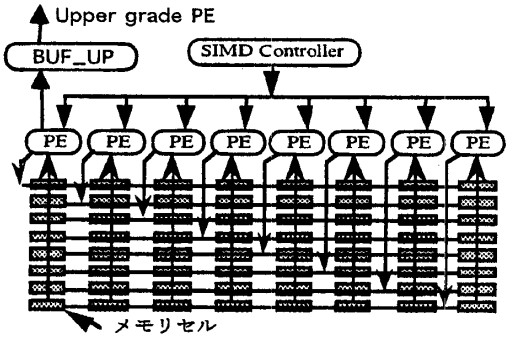

図３．SIMD遺伝アルゴリズム専用プロセサの構成

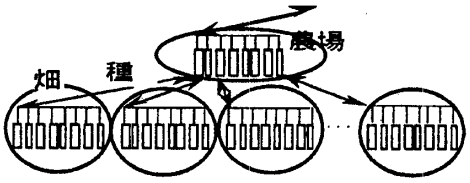

図４．Farming Modelによるシステムの階層化

表１．SMMの各セルの構成

個体	32bit	roulette	12bit
total value	9bit	value(4 item)	16bit
total weight	9bit	weight(4 item)	16bit

valueの領域に，適応度はrouletteの領域に書き込まれる．

初期集団の生成： 乱数レジスタRNDは常に新しい乱数を表示しており，初期個体はRNDから３２ビットの乱数を２個取り出し，SMMに書き込む．

ソーティング： 評価値により個体をＯ（ｎ）のソーティングを行う．個体の評価値によりソーティングを行うので，Farming Modelにおいて良い種の収穫に適している．

淘汰(Selection)： 乱数により適応度から作成したルーレットを回すことにより，次の親となる個体を選択する．適応度の大きいものほど選択される確率が高く，低いものは淘汰される．二分検索法を用いて行う．

交差（Crossover）： ２点交差，単純交差，一様交差を行うことができる．交差率と乱数により交差を行うかを決定し，交差を行う．２点交差，単純交差は専用回路を用いてXOVR命令により行う．

突然変異（Mutation）： 突然変異率と乱数により突然変異を起こすか否かを決定し，乱数によって求めた特定のビットを反転することにより突然変異を起こす．

５．ＧＰＭとＦＭのシミュレーション結果

Global Population Model (GPM)とFarming Model (FM)を用いて，ナップザック問題に適用し，シミュレーションを行い収束性の検討を行った．個体数64，アイテム数は32，最大価値，最大重量はともに9，ナップザックの容量を80，交差率0.9，突然変異率0.2とし50回の試行を行い，最適解が出現した世代を求めた．GPMでは64台のプロセサを用い，個体は１つの共有メモリに集められているとし，FMでは畑の個体数，畑の数をともに８とし，農場での集める種の数を８，したがって，８＊８＝６４個の個体で，農場ではこれらの個体を用いて遺伝操作を８台のプロセサで行う．

シミュレーションの結果，GPMは平均５０世代で，FMでは９７世代で最適解が得られた．図５にそれぞれの収束した世代数と頻度を示す．

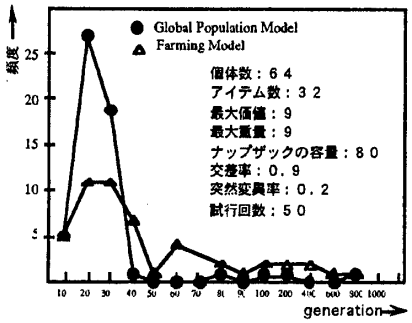

図５．GPMとFMの収束性の比較

この結果，GPMと比較してFMでは最適解を得るまで２倍程度世代交代が必要となる．演算速度は，GPMでは共有メモリを使用しているので，６４回の遺伝操作に必要な共有メモリへのアクセスがクリティカルパスとなり，一世代の演算時間が大きくなる．一方，FMでは，SWMR（Single Write Multiple Read）型メモリであるSMMを用いることにより，全てのプロセサは同時に共有メモリにアクセスでき，クリティカルパスは１となり，GPMと比較して１/６４の時間で演算できるので個体数に関係なく短時間で演算できる．この結果，FMを用いたSIMD型並列計算機が個体数の大きい遺伝アルゴリズムに適することが明らかになった．

６．まとめ

FM方式によるSIMD型遺伝アルゴリズムを提案し，GPMとFMの収束性をシミュレーションにより検証した．遺伝アルゴリズム専用PEを設計した．また，Farming Modelによる階層化により，探索問題の個体数に影響されない演算速度が得られることが確認された．現在，FPGAを用いてSIMD型並列計算機を製作中で，今後実機を製作して実験を行う．

参考論文

[1] Lawrence Davis: Handbook of Genetic Algorithms, Van Nostrand, pp.1-22, 1991
[2] D. E. Goldberg : Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, pp. 208-212, 1989
[3] 北野宏明編：遺伝的アルゴリズム，産業図書，pp.1-66, 1993
[4] 高橋義造編：並列処理機構，pp. 199-206, 丸善，1988
[5] Hennessy , Patterson （富田眞治訳）：コンピュータ・アーキテクチャ，日経BP出版センター，pp. 191-235, 1992
[6] Kai Hwang : Advanced Computer Architecture, McGraw-Hill , pp. 238-247, 1993
[7] Tanese, R : Distributed Genetic Algorithm , Proc. of 3rd Int. Conf. on Genetic Alogorithms, pp.434-439, 1989
[8] Lawrence Davis: Genetic Algorithms and Simulated Annealing, Morgan Kaufmann, pp. 129-140, 1987
[9] 井上，稲富，佐野，高橋：SIMD型遺伝アルゴリズムと専用プロセサ要素の設計，情報処理学会研究報告，ARC116, pp.43-48, 1996