並列分散ＧＡ

北野　宏明

本論文では，遺伝的アルゴリズム（ＧＡ）と並列処理に関して議論する．ＧＡは，非常に並列度を高くすることが可能な手法であり並列処理によって高速化を行なうのに適している．多くの場合，各個体の評価は独立に実行することができるため，プロセッサを増やすことによってはほぼ線形な速度向上が達成され得ることが知られている．ここでは，ＧＡを並列化することの利点を議論するとともに，筆者らが構築しているＧＡ-１システムとその設計理念について議論する．

1. はじめに

現実問題に，ＧＡが応用され，しかも実時間で結果を返さなければならないような分野に使用される場合を考えてみよう．ここで問題となるのは，１世代の評価に，個体の数だけ評価時間がかかることであり，１個体に対する評価時間が大きいときには，実時間レスポンスは，困難となる．しかし，ＧＡも分類システムも，非常に並列度の高い計算機構であるので，当然のことながら，並列マシンの上に実装するという方法がある．ここでは，ＧＡ自体がスキマタに対してもっている内在的並列度（Implicit Parallelism）ではなく，個体評価や分類子のマッチングなどの陽に現われる並列性の議論に焦点を当てる．

並列分散ＧＡには，基本的に２つの考え方がある．１つは，いくつも下位集団（subpopulation）をもつことによって局所解の全集団への伝搬を避けよりよい解を得ようとする考え方．もう１つは，並列処理による高速化をねらう考え方．しかし，どちらか一方のみを考えることは少なく，実際には解の質の向上と高速化を同時に得ようとする．

解の質の向上に関しては，Tanese による研究がある[Tanese, 1989]．Tanese は，Walsh 多項式を使って分散ＧＡと従来型のＧＡでの解の質を比較した．分散ＧＡとして，複数の下位集団があり，その間で移住（Migration）が定期的に発生するモデルと，単一集団との比較を行なった．その結果，分散ＧＡは，一貫して従来型の単一集合によるＧＡより質の高い解が得られることがわかった．これは，分割されたモデルでは，局所解が全体に広がることが抑制され，各下位集団ごとに違った局所解をサンプルすることが可能になるためであると思われる．

一般に，小さな集団では，局所解に速く収束し，その後適応値は向上しない可能性が高い．しかし，分散ＧＡのように，ある一定の間隔で移住を行なうことにより，局所解にすべてが収束してしまう危険を避けることができ，進化の停滞を防いでいる．

また，超並列マシンへの実装としては，Robertsonが，Connection Machine に分類システム *CFS を実装し１サイクル５ミリ秒の値を得ている事例[Robertson, 1987]や，トランスピュータを64個トーラス状に結合したマシン[Gorges-Schleuter, 1989]，さらには，連想メモリの使用[Twardowski, 1990]など，最近は非常に活発に行なわれれいる．ここでは，筆者らが開発しているＧＡ－１を例にＧＡの並列化について議論する．

2. ＧＡ-１

GA-１は，電総研の樋口らが開発した，IXM-２連想メモリ・マシン[Higuchi et. al., 1990]を基盤として開発され，並列ＧＡ開発・実行環境である．GA-１は，実時間レスポンスが要求されるような，現実問題に対してＧＡや分類システムを使用しようとするときに用いられることを意図して開発された．また，ＧＡ研究の環境としても，いろいろなパラダイムの実験が可能であり，非常に有用なシステムである．

GA-１の特徴は，連想メモリを装備したことにより，ビット・ベクトルで表現された分類子のマッチングが非常に高速で行なわれることであり，ＧＡを利用したルール獲得のシステムの研究・開発には最適といえる．GA-１では，256K個の分類子の並列マッチングが可能であり，これによって，コネクション・マシンより２桁近く速い実行速度を達成している．このため，ピッツ・アプローチ[Smith, 1980]，やミシガン・アプローチ[Holland

きたの　ひろあき　日本電気㈱
〒108 港区芝浦２－11－５

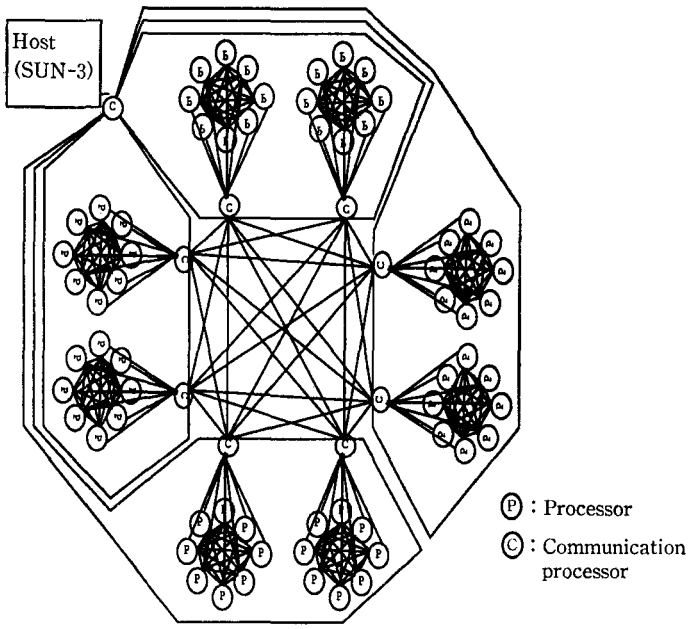

図 1　IXM2連想メモリ・プロセッサの構造

表 1　ノード間の平均距離

No. of PE	IXM 2	hypercube	torus
4	1	1.33	1.33
8	1	1.71	——
16	2.06	2.13	2.13
32	2.54	2.61	——
36	——	——	3.08
64	2.77	3.04	4.03

and Reitman, 1978]が，ハードウェア・レベルから直接サポートされることになる．

2.1　IXM2連想メモリ・プロセッサー

IXM 2は，電総研で開発された超並列連想メモリ・プロセッサである[Higuchi, et.al., 1990]．IXM 2は，64個の連想プロセッサと9個の通信プロセッサから構成されている．連想プロセッサは，T800トランスピュータと1ワードが40ビットの4Kワードの連想メモリを中心に構成されている．通信プロセッサも，T800トランスピュータを使用している．ノード間結合は，階層的完全結合を使用している．

表1には，いろいろなコネクション・トポロジーでのノード間の平均距離を示した．IXM 2は，完全結合であり，64ノードまでは，どの結合方式よりも短い距離でノード間が結ばれている．このような完全結合方式は，実時間応用に重要である高いバンド幅をもたらすのである．各々の結合は，高速シリアル・リンクで，20Mbits/sec の速度を達成している．これによって，最大 2.4 Mbytes/sec のデータ転送が可能となる．

2.2　並列マシン上へGAを実装する問題点

ここでは，GA-1の設計に際して考慮した点を議論しよう．ここでとりあげる問題は，(1)ルール解釈の高速化 (2)ノードへの集団のマッピング，(3)選択交差と世代モデル，である．

2.2.1　ルール解釈の高速化

今まで，ルール・ベース・システムとして研究されてきた方法[Holland and Reitman, 1978] [Robertson, 1987] [Barto, et. al., 1985]やピッツ・アプローチとして研究されてきた方法[Smith, 1980] [Grefenstette, et. al., 1990] [Greene, 1987]は，いずれも，固定長のバイナリ表現を使用するという特徴を有している．GA-1では，連想メモリを使用することによって，固定長バイナリ表現されるルールの探索を高速化している．これは，実時間大規模GAシステムを開発する際には非常に重要になる．特に，CS-1やLS-1の系統をひきつぐルール学習のシステムを作るには，重要なファクターである．GAにもとづくルール学習では，染色体はルールの集まりまたはルール集合の集まりである．各々のルールには，条件部と実行部がある．連想メモリの上に表現する際には，Pit アプローチかミシガン・アプローチかの違いは問題にならない．このような，方式の違いは評価や確信値分配，や選択交配などのレベルで問題になるが，ルール探索では関係はない．基本的に，条件部が連想メモリに格納され，実行部はRAMに格納される(**図2**)．各ノードの連想メモリは，条件部40ビットのルールを 4,000 ルール格納できる．64ノードをすべて使用すると，256Kルールを並列探索することができる．

ルールの探索をする際には，サーチ・マスクとサーチ・データが必要になる．サーチ・マスクは，どの部分のメモリーが検討されるべきかを指示する．次ページの下段に，プログラムの一部を示す．

2.2.2　集団のマッピング

各ノードに，個体または集団を割り当てる方法にはいくつかある(**表2**)．まず最初に，1個体を1つのプロセッサに割り当てることができる．これは，各々の個体の

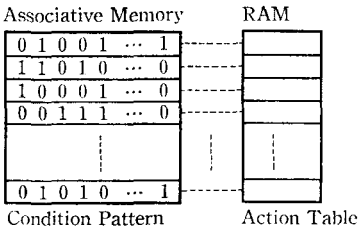

図 2　連想メモリ上でのルール表現

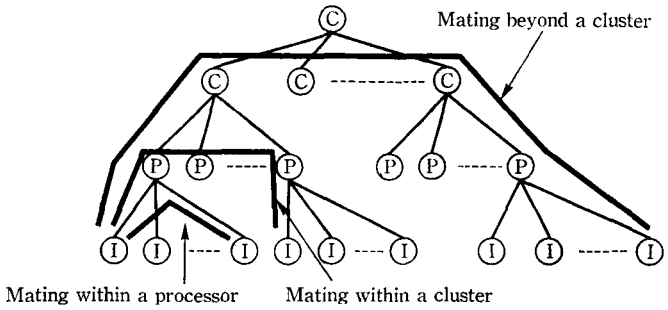

図 3　3：GA-1 上での交配と移住

適応度の評価に大きな計算量が必要なときに有効な方法である．この場合，64ノードのGA-1では，集団は最大64の個体となる．いくつかの下位集団を生成するなら，たとえば，8ノードを一集団として，8個体の集団を，8集団作ることもできる．

また，違う方法では，各ノードに1集団を割り当てることもできる．この場合では，64集団まで実装できる．

どのように集団や個体がノードに割り当てられるかによって，選択交配や移住等の戦略が変わってくる．

GA-1は，選択交配のレベルとして3つのレベルをもっている．(1)ノード内，(2)クラスタ内，と(3)クラスタ間である．これは，図3に示した．集団モデルと最大交配距離の関係を表2に示した．

ピッツ・アプローチを使用するときには，基本的に1プロセッサに1個体を割り当て，全体で1つの個体集合とする．この場合，各々最大4Kルールをもった64個体が並列に評価される．ミシガン方式では，集団または下位集団が各ノードに割り当てられ，全体で複数の集団をもつシステムとなる．これによって，複数の分類システムが並列に評価される．これらの割当方法以外にも考えられる．また，各集団を違う種としたり，共生進化のモデルを作ることも可能である．

2.2.3　選択交配と世代モデル

表 2　GA-1 上での集団の構成法と選択交配の範囲

集団構成	プロセッサへのマッピング	最大交配範囲
単一集団	個体	クラスタ間 クラスタ内
	下位集団	クラスタ間 クラスタ内
複数集団	個体	クラスタ内
	下位集団	クラスタ間 クラスタ内
	集団	プロセッサ内

どの選択交配モデルと世代モデルを使用するかは，プロセッサのアイドルや交配時の通信ボトルネック等によって処理速度に関係してくる．一般的に使われている，中央で制御する方法を，離散型世代モデルと組み合せて使う方法はできれば避けたい方法である．

まず，第1に，個体評価の計算量が各個体で違ってくるときには，プロセッサ・アイドルが不可避的に発生する．後で行なう計算では，80%以上の全CPU時間が無駄になるという結果になる[Kitano, 1991]．

第2に，同期型の選択交配では，すべての個体間の染色体の交換が終了するまで次の評価が開始されない．この染色体の交換が同期して発生するため，メッセージ衝突が大量に発生することになる．各々の個体の評価時間

```
AM.Search.Mask:=literal.search.mask -- specify bits to be searched
AM.Search:=literal.search.data      -- initiate search operation

classifier[0]:=AM.Read.IN          -- retrieve one classifier just hit
address:=AM.Addr.Reg /\ #FFFF      -- get the address of the classifier
WHILE (address <> End.OF.Hit)      -- repeat until all the matched classifiers
  SEQ                              -- are retrieved
    classifier[i]:=AM.Read.IN
    address:=AM.Addr.Reg /\ #FFFF
    i:=i+1
```

プログラムの一例

が比較的短い場合には，この通信の問題は見逃すことはできない問題になってくる．局所的な交配，たとえば，近傍の個体のみとの交配，では通信の問題は大幅に軽減される．ただし，この場合にも，プロセッサ・アイドルの問題は残ってしまう．

GA-1では，必要があれば全体的な同期にもとづく選択交配を行なうことはできるが，主には，連続世代モデルの使用を推薦している．連続世代モデルでは，各々の個体が非同期的に選択交配を行なう．交配の相手の範囲は全体からでも近傍からでもよい．各個体の評価が終了した時点で，非同期的に交配を可能としたことで，プロセッサ・アイドルは大幅に軽減される．

また，連続世代モデルは同期をしないので，メッセージ衝突の確率は大幅に低減する．

2.3 GA-1でのルール学習

GAでのルール学習は主に2つのパラダイムから研究されてきた．1つは，分類システムまたはミシガン・アプローチという方法で，もう1つがピッツ・アプローチという方法である．すでに，述べたようにこれら2つの方法は，ルールを個体と見るか，ルール・セットを個体と見るかの違いがある．

分類システムは，ルールの集合を問題解決の直接の方法としており，GAは，新たなルールを生成する手段として位置づけられている．ルールの評価などは，バケット・ブリゲート・アルゴリズム等で行なわれる．ピッツ・アプローチでは，よりGAを最適化に使用する時に近い手法となっている．各個体は，ルール集合であり，GAは，このルール集合を最適化するために使われる．

タスク指向の考え方で見てみると，分散システムは，主に，自律エージェントの見方から始まっている．それは，環境とインターラクトしながら問題解決を行ない，実時間で知識ベースを漸増的に変化させるという手法である．ここで重要なのは，ルール・セットが，連続的に変化することであり，飛躍的な変動は望ましくない．この方式を並列化する利点は，反応の速さであり，対応できる問題の複雑性が大幅に高くなることである．

ピッツ・アプローチはどちらかというと，Off-line的手法である．トレーニング段階で，ルール・セットを獲得し，実行時は，与えられたルールを高速実行する．現在あるデータの規則性の抽出をして予測にしようとすることや，どのシミュレーション・モデルが問題解決に適しているかを判定する問題 [Grefenstette, et. al., 1990] では，ピッツ・アプローチは最も自然な方法である．ここでは，並列化は，トレーニング時間の短縮となって効果を表わす．これは，さらに大規模かつ複雑な問題に挑戦できることを意味するし，さらに大きな集団でより徹底したトレーニングを行なえるようになる．

2.3.1 分類システムの並列化

どちらの方法においても，逐次処理マシンでの実装時に問題となるのはルール解釈時の処理時間である．すでにGA-1では，連想メモリーを使用することによってこの問題を回避できることを述べたが，連想メモリーを使用するというアイデアは，すでにTwardowskiが簡単な分類システムをその上で開発している [Twardowski, 1990]．彼の，Coherent Processorは4Kの連想メモリを実装している．GA-1では64ノードなので，はるかに大きなルール・セットを使用できる．実際，GA-1では，CM-2 Connection Machine での分類システム [Robertson, 1987] と同等の大きさのルール・セットがより高速に評価できる．

より重要な問題，つまり分類システムの学習の問題に関しては，Twardowski が開発した選択交配の手法がGA-1の場合にもそのまま当てはまる．しかしながら，GA-1の場合には，もっと複雑な分類システムを並列化することも可能になっている．それは，[Holland and Reitman, 1978] で示されたように，複数のルール・セットをもち，各々違った外界の信号・刺激に対して適応するようなものである．

2.3.2 ピッツ・アプローチの並列化

ピッツ・アプローチの並列化は，分類システムの並列化よりさらに計算量的にシビアな手法である．これは各々の個体がルール・セットであるために，個体分の適応度評価を行なわなければならないからである．Grefenstetteは，SAMUEL システムを128ノードの BBN Butterfly Machine に実装して高速化を計っている．ここでは，全体域選択交配と同期型の世代モデルが使われたが，個々の個体は並列に評価されている．評価時間は，逐次型に比べて2桁以上短縮されているが，それでも適応度の評価が性能上の問題点となっている．

同じような実装はGA-1の上でも可能である．各々のノードに個体を割り当てれば，64までの並列性が確保できる．BNN Butterfly の128に比べて，ノードレベルの並列度は半分になっているが，実際には，ルール解釈に連想メモリが使用されるために，GA-1の方が高速にSAMUELシステムを実行できることが予測できる．実

際問題，GA-1では，個体評価より，同期や通信の方が問題になるであろう．

これは，基本的には，各々のタスク・ドメインに依存することである．たとえば，ANIMATやCS-1 [Holland and Reitman, 1978]の地図作成のドメインでは，評価時間は個体の適応度と反比例の関係にある．このような場合には，連続世代モデルで，プロセッサーのアイドル時間を低減できるだけでなく，実際に有効な選択交配の数を増やす方向になる．

適応度と評価時間が正比例するようなドメインでは，プロセッサ・アイドル時間は減少させることができるが，全体の選択交配での相手の選択に制約が発生する．このような，タスクの代表的な例がSAMUELのevasive maneuver domainである．

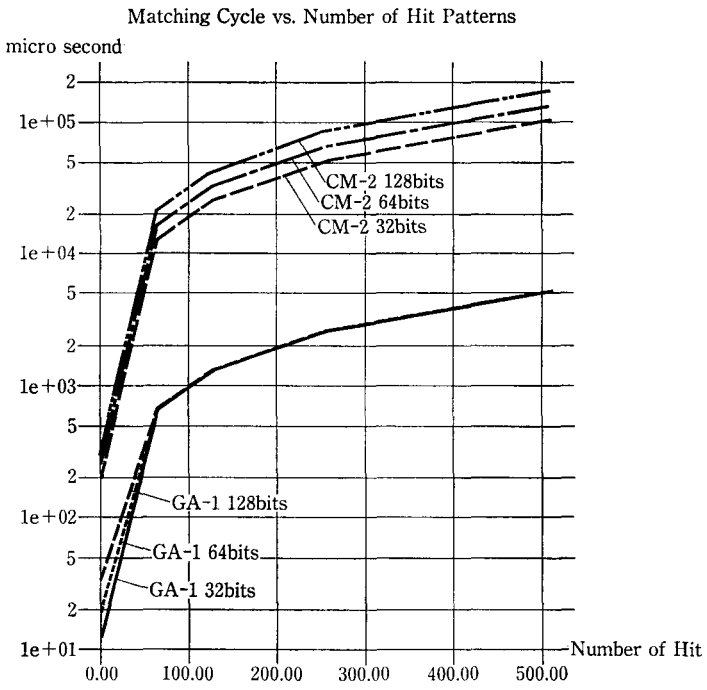

図 4　GA-1の性能(1)

ピッツ・アプローチをGA-1上で並列化する際に，アーキテクチュアからの制約がある．まず，実際問題として，各プロセッサに付随するメモリ領域の一部を個体評価のプログラムとトレーニング・データに割り当てる必要がある．これは，それほどトレーニング・データを使う立場からみれば，さほど重要な制約ではない．かなりのトレーニング・データを格納しても，相当な数のルールを表現するだけの領域はある．しかしながら，シミュレータを使う場合には，各プロセッサに128 Kバイトという制約は大きく，実世界に近いような精巧なシミュレーションは難しいであろう．一般的に，GA-1は，トレーニングデータが事前に用意できるようなタスクに向いている．場合によっては，[Greene, 1987] にあるようなウインドウを使うのもよい．とはいうものの，現在までのほとんどのシミュレーションを用いた研究のサイズならGA-1にそのまま実装できる．また，ノードに実装されるメモリーを増やせばこの問題は解決される．

第2の問題点は，プロセッサの数に関係するものである．つまり，最大の個体数が64というのは，[Grefenstette, et. al., 1990] らの研究を見る限りにおいては少し少ないかもしれない．

2.4　性能評価

ここでは，GA-1の基礎的な性能データを見てみる．

図4は，連想メモリに格納されたルールの照合・検索時間を示している．照合検索時間は，ルールの条件部の長さに依存する．これは，長い条件部の探索には，複数回連想メモリの照合を行なう必要があるからである．しかしながら，この時間は，ルールの数に対しては変動しない．照合は，並列であるが，検索はどうしても逐次的に行なわざるを得ない．検索で得られる並列度は，64が最大である．条件部が40bitsであるとき，1回のマッチに0.125マイクロ秒かかる．もし，256Kルールが GA-1全体に格納されているなら，逐次型で同じ速度を達成するには，各ルールをナノ秒以下で照合できなければならない．

条件部が40ビットであるルールを使い，1サイクルでのヒット数が10程度とする．この場合，GA-1は，1秒間に7,700サイクルを達成する．これは，Connection Machine と比べても桁違いに早い値である．(10cycles per seconds with 65,000 classifiers [Robertson, 1987]).

CM-2を使って，同じ条件で実験してみると，CM-2では，最低68.7 milliseconds かかる．これは，すべて

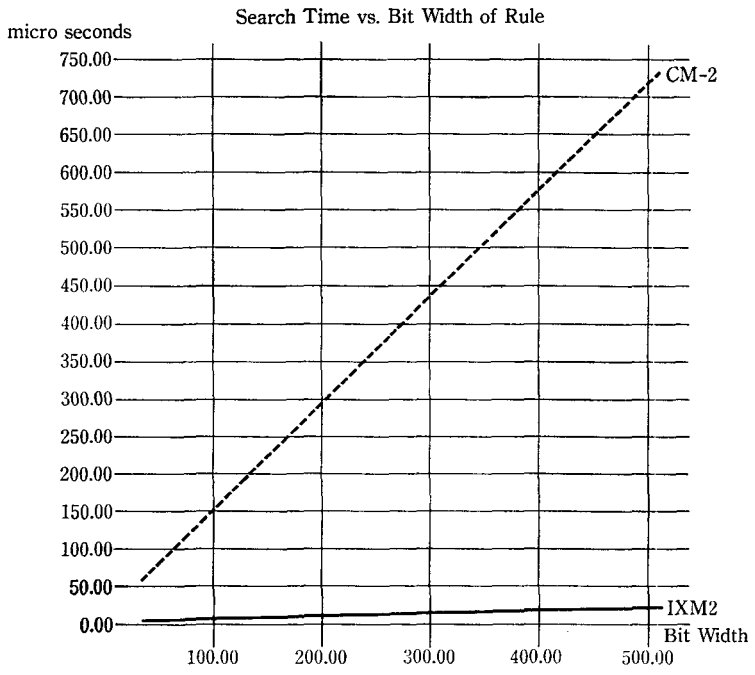

図 5 GA-1の性能(2)

のプロセッサのヒット判定のデータをホスト側に，検索する必要があるためである．ここでは，プロセッサ・ホスト間の通信がパフォーマンスを阻害している．つまりマッチした結果の検索に時間がかかっているのである．また，CM-2では，マッチしたかどうかのチェックを，すべてのノードをチェックするので，いくつのノードがマッチしたかと通信の量は関係ない．多くの場合，少ないルールがマッチするので，これはかなりのオーバヘッドになる．CM-2でのマッチ時間は，条件部の長さに比例する．（これは，**図5**に示されている．）ただし，この時間では，通信のオーバヘッドに比べれば微々たるものである．逐次型マシンで，GA-1の性能を出すためには1秒間に19億7100万回のビット・ベクトル・マッチを行なわければならない．これは，現在のデバイス技術では到達不可能な値である．

図5には，条件部の長さごとのマッチ時間を示した．GA-1は，基盤となるIXM-2の連想メモリをフルに活用しておりCM-2より2桁近く早い値が出ている．

非常に大きなバンド幅（2.4Mbyte/second）のため，染色体の交換で実行時間が大幅に遅延することはない．あるノードの連想メモリに格納された染色体を他のノードに転送するのに10.5マイクロ秒しかかからない．最悪の場合，つまりすべてのノードが1つのノードに染色体全部を送るような場合を想定しても，75マイクロ秒程度しかかからない．これは，実時間タスクに使用しても十分な速度である．しかし，GA-1は，マッチ・サイクルが非常に高速であるため，この程度の通信時間もその最高性能を引き出す障害になることがある．つまり，GA-1でのマッチ・サイクルは，数マイクロ秒であるので，75マイクロ秒は，GA-1では，数マッチ・サイクルに相当する．1世代での，各々の個体の評価が数マッチ・サイクルで完了するタスクは十分考えられる．そのような場合では，同期型の染色体交換は，性能悪化の原因となる．このような場合には連続世代型GAを導入するのがよい．

3. おわりに

GA-1においては，連想メモリの使用が分類システム実行の際の高速化の決め手になったわけであるが，他のモデルを実装する際には，別のアーキテクチュアが優位になることは考えられる．しかしながら，GAの並列化に関して，いくつかの基本的事項をふまえる必要がある．

- 各個体評価の方法とプロセッサ能力の関係——シミュレータを評価に使用する際には，それなりに強力なプロセッサにタスクを割り当てることが必要．
- 分類システムでは，ルール照合の高速化——連想メモリの使用
- 世代モデルの選択——連続世代モデルなどを導入し，プロセッサ・アイドルを減少させる．
- 近傍モデルや集団構造の選択——交配の範囲や下位集団構造の選択．

基本的に，並列マシンの計算能力を最大限に発揮させ解発見の時間を短縮する設計と，分散的集団構造を使用し解の質を向上させるという，2つの戦略が基本となる．最適な，実装方法やハードウェア化は，それらを勘案しながら決定されるべきものである．これらの基準からGA-1を見ると，分類システムを中心とした，ルール学習，実時間システムには，最適の構成となっている．

参考文献

[Barto, et. al., 1985] Barto, A., Anandan, P., and Anderson, C., "Cooperativity in networks of pattern recognizing stochastic learning automata," *Proceedings of the Fourth Yale Workshop on Applications of Adaptive Systems*, 1985.

[De Jong, 1975] De Jong, K., *An analysis of the behavior of a class of genetic adaptive systems*, Doctoral dissertation, University of Michigan, 1975.

[Fitzpatrick and Grefenstette, 1988] Fitzpatrick, M., and Grefenstette, J., "Genetic Algorithms in Noisy Environments," *Machine Learning* 3, 2/3, 1988.

[Gorges-Schleuter, 1989] Gorges-Schleuter, M., "ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy," *Proceedings of ICGA-89*, 1989.

[Greene, 1987] Greene, D., Smith, S., "A Genetic System for Learning Models of Consumer Choice," *Proceedings of ICGA-87*, 1987.

[Grefenstette, et. al., 1990] Grefenstette, J., Ramsey, C., and Schultz, A., "Learning Sequential Decision Rules Using Simulation Models and Competition," *Machine Learning*, 1990.

[Higuchi, et. al., 1990] Higuchi, T., Furuya, T., Kusumoto, H., Handa, K., and Kokubu, A., "IXM2 : A Parallel Associate Processor for Semantic Network Processing," *Proceeding of the International Conference on Tools for Artificial Intelligence*, 1990.

[Hillis, 1985] D. Hillis, *Connection Machine*, MIT Press, 1985.

[Holland and Reitman, 1978] Holland, J., and Reitman, J., "Cognitive systems based on adaptive algorithms," Waterman, D., and Hayes-Roth, F., (Eds.) *Pattern directed inference systems*, New York, Academic Press, 1978.

[Inmos, 1987] Inmos, *IMS T800 Transputer*, April 1987.

[Kitano, 1990a] Kitano, H., "Empirical Studies on the Speed of Convergence of Neural Network Training by Genetic Algorithms," *Proc. of AAAI-90*, 1990.

[Kitano, 1990b] Kitano, H., "Designing Neural Networks Using Genetic Algorithms with Graph Generation Systems,"*Complex Systems*, Vol.4, No.4, 1990.

[Kitano, 1991] Kitano, H., "Continuous Ceneration Genetic Algorithms," *J. SICE* (計測と制御), Vol. 32, No.1, 1993.

[Ogura, et. al, 1989] T. Ogura, J. Yamada, S. Yamada and M. Tanno, "A 20-Kbit Associative Memory LSI for Artificial Intelligence Machines," *IEEE Journal of Solid-State Circuits*, Vol.24, No.4, August 1989.

[Robertson, 1987] Robertson, G., "Parallel Implementation of Genetic Algorithms in a Classifier System," Davis, L., (Ed.) *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, 1987.

[Smith, 1980] Smith, S., *A Learning System Based on Genetic Adaptive Algorithms*, Ph. D. Thesis, University of Pittsburgh, 1980.

[Tanese, 1989] Tanese, R., "Distributed Genetic Algorithms," *Proceedings of ICGA-89*, 1989.

[Twardowski, 1990] Twardowski, K. E., "Implementation of a Genetic Algorithm based Associative Classifier System (ACS)," *Proceedings of International Conference on Tools for Artificial Intelligence*, 1990.