人 工 知 能 77−20−5
（1991. 7. 18）

遺伝アルゴリズムと並列処理

丸山 勉、小長谷 明彦、河野 秀樹†、小柳 敏†、山岸 晃一†、小西 弘一
日本電気(株) C&C システム研究所　†日本電気技術情報システム開発(株)

1 はじめに

近年、ヒトをはじめとして様々な生物の遺伝子情報(DNA 配列情報、アミノ酸配列情報)が分子レベルで収集されつつあり、これとともに、計算機を利用した遺伝子情報解析技術への期待が高まっている。このような遺伝子情報解析技術の一つとして、配列情報からの規則(モチーフ)抽出がある。我々は、これまで、遺伝子情報が生物種の多様性に由来するノイズを含むことに着目し、モチーフ抽出を確率的規則の学習問題として定式化し[1]、遺伝子情報に向いた確率的規則の表現形式として確率的決定述語を提案し[2]、より良い確率的決定述語の選択基準として記述長最小(MDL)基準を利用するモチーフ抽出法を提案してきた[3]。MDL 基準を用いたモチーフ抽出法では、モチーフの良さをモチーフを表現する確率的決定述語の記述長とモチーフの正確さを表す記述長(確率的決定述語の対数尤度)の和(少ないほど良い)で表す。したがって、与えられた配列情報について全ての確率的決定述語の記述長を計算すれば、原理的には計算機による自動抽出を行なうことができる。

モチーフの自動抽出において、抽出されたモチーフの良否は考慮すべき確率的決定述語の集合、すなわち、仮説空間の設定の仕方と、仮説空間内での探索アルゴリズムの両方に依存する。モチーフ抽出の場合、仮説空間を事前に絞り込むことが困難なこと、必ずしも最適解を求める必要がないことから仮説空間を十分大きくとり、確率的探索アルゴリズムにより準最適な確率的決定述語を求めるという方針を採った。また、確率的探索アルゴリズムとして遺伝アルゴリズムを採用した。本稿では、この遺伝アルゴリズムを並列化した際の挙動に関して知見を得たのでこれを報告する。以下、はじめに、2節において遺伝アルゴリズムの基本的な考え方を紹介し、3節で本稿で採用した並列遺伝アルゴリズムについて述べる。また、4節で並列マシン上での実験結果について報告する。

2 遺伝アルゴリズム

遺伝アルゴリズムは、生物の進化の過程をモデルとして発案された確率的探索アルゴリズムの一つである。その特徴は、仮説空間内の候補に0と1のビット列を対応させ、このビット列を遺伝子と見立てて増殖、交差、突然変異などの遺伝子操作を加え、より良い解へ進化させることにある。例として、アミノ酸配列からのタンパク質のモチーフの抽出を考えよう。

motif(S,cyto_c) with p_1
:- contain(S,"CXXCH").
motif(S,others) with p_2.

上記の確率的決定述語は、アミノ酸配列 S がパターン "CXXCH"(X は任意のアミノ酸と照合する)を含めば確率 p_1 でシトクロム C であり、そうでなければ確率 p_2 でその他である(確率 1-p_2 でシトクロム C である)という確率的モチーフを表す。今、このような確率的決定述語のパターンの候補として、

"CXXCH", "GXLXG", "PGTKM"

の3つがあるとする。この場合、各々のパターンは独立に現れるとすれば、遺伝子としては3文字のビット列を考えればよい。各ビットが1であるとその対応するパターンが含まれることを意味する。遺伝アルゴリズムによる解の探索は以下の手順で行なわれる。

1. 初期化: ランダムに遺伝子を選ぶ
   100, 011, 010
2. 増殖: 遺伝子を適応度に応じて増殖させる。
   100,011,010 の適応度が 2,1,0 ならば以下のようになる
   100, 100, 011
3. 交差: 2つの遺伝子の間で N ビットを交換する
   例えば 2,3 番目の遺伝子の 2~3 ビットを入れ換える
   100, 111, 000
4. 突然変異: ビットをランダムに反転させる
   例えば 3 番目の遺伝子の 3 ビットを反転する
   100, 111, 001

以下 2. から 4. の処理を N 回繰り返し、N 世代目の遺伝子を得る。

3 並列遺伝アルゴリズム

遺伝アルゴリズムにおいては、各遺伝子間の独立性が高いことから、並列化による処理の高速化が期待できる。並列遺伝アルゴリズムの一つとして以下の方式について実験を行なった。

1. 初期化において幾つかのグループに遺伝子を分割する。
2. 前節の処理を各グループに対して行なう。
3. 突然変異の後にグループ間で遺伝子を幾つか交換する。

逐次的に行なった場合との違いは、交差および増殖が各グループ内で行なわれることにある。交差については、グループ間で適当に遺伝子の交換を行なっているので全体を一つのグループとして扱った場合とそれほど差異がないと考えられるが、増殖の場合には、(N/グループ数)個の中でのみ生存競争

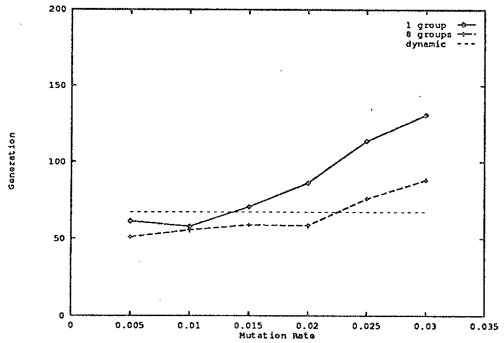

図 1: 最適解が求まるまでの世代数

表 1: 最適解が得られなかった回数 (100 回中)

突然変異確率	0.005	0.01	0.015	0.02~0.03
1 グループ	2	0	0	0
8 グループ	10	3	1	0

が行なわれることになるため遺伝子の多様性が失われやすくなる可能性がある。多様性を保つためには突然変異確率を大きくすればよいが、一般に (準) 最適解を得るまでの世代数は長くなる。そこで、各グループ毎に異なる変異確率を用意し、適当な間隔でグループ間で遺伝子の評価を比較し、よりよい評価を得たグループの変異確率に他のグループの変異確率を少しずつ近付ける方法を試みた。次節にその結果を示す。

4 評価結果

確率的決定述語のパターンの候補として 40 パターンがある場合について実験を行なった。128 個の遺伝子 (40 パターン候補があるので長さ 40 ビットとなる) をランダムに用意し以下の 3 種類の場合について最適解が求まるまでの世代数を測定した。

1. 全ての遺伝子を 1 つのグループ
   (突然変異確率一定)
2. 遺伝子を 8 つのグループに分割
   (a) 突然変異確率が同一でかつ一定
   (b) 突然変異確率の初期値はグループ毎に異なり、それらを動的に変更

突然変異確率の動的な変更は、各グループに初期値として突然変異確率 0.005 + 0.004n (n=0~7) を与え、ある世代間隔で各グループ内の遺伝子の評価値 (MDL 基準による記述長) の平均を隣のグループと比較し、悪い方のグループはよい方のグループにその突然変異確率を 50% 近付けるという方式を用いた。

100 回の測定を行ない最適解が得られたときの世代数の平均値を図 1に、100 回のうち最適解が得られなかった回数を表 1に示す。図 1において交差確率は 1 である (1 の場合が解が最も速く求まる)。

全般に並列化した場合の方がより早い世代で最適解が得られいる。また突然変異確率が小さい程、最適解が得られるまでの世代数が早い。しかし、表 1に示したように突然変異確率が低いと最適解が得られない危険性も高くなる。並列化を行なった場合には、特にこの傾向が激しい。これは、並列化した場合には、1 グループ中の遺伝子数が少ないことによって遺伝子の多様性が失われたためと考えられる。動的に突然変異確率を変えた場合の値は図 1で横線で示されている。この方式では 100 回とも全て最適解が求まっており、かなり良い結果が得られていることがわかる。

処理の並列化による速度向上は、8 プロセッサを用いた場合で約 7.7 倍である。アルゴリズム的にみてもほとんど並列化のオーバーヘッドがないため、より多くのプロセッサを用いればより高速な処理が実現できると考えられる。

5 まとめ

遺伝アルゴリズムにおいて、並列化が処理速度の向上に著しく有効であることが確認された。並列化を行なうと遺伝子の多様性が失われやすくなるため、最適解が求まらなくなる危険性が高まるが、各グループの変異確率を動的に変更する手法によって、このような危険性を回避し得ることがわかった。

より並列化するためには各グループ内の遺伝子数をより少なくすることが必要であり増殖の過程にも、並列化のための工夫が必要となる。今後、これらの点を中心に遺伝アルゴリズムの並列化の研究を進める予定である。

謝辞

本研究の機会を与えた下さった ICOT 7 研新田室長に深謝致します。

参考文献

[1] yamanishi:91Yamanishi, K. & Konagaya, A.(1991). Leaning Stochastic Motifs from Genetic Sequences. *to appear in the Eighth International Workshop of Machine Learning.*

[2] konagaya:91Konagaya,A. & Yamanishi, K. (1991). A Stochastic Desicion Predicate: A Scheme to Represent Motifs, to appear in the AAAI Workshop of Classification and Pattern Recognition in Molecular Biology.

[3] 小長谷, 山西,(1990). 「記述長最小基準の遺伝子情報処理への適用について」, ソフトウエア科学会第 7 大会論文集,pp.101-104.