計算機アーキテクチャ 115－5
設 計 自 動 化 78－5
（1995. 12. 14）

遺伝的アルゴリズムを用いたスレッド分割法

片山 清和, 市川 周一, 島田 俊夫

名古屋大学工学部
〒464-01 名古屋市千種区不老町

あらまし

プロセッサの演算実行部の使用率を向上させるために、プログラムを複数のスレッドに分割し、それらを同時に実行する方法が提案されている。プログラムをスレッドに分割する手法としては、経験的手法を用いる方法と遺伝的アルゴリズムを用いる方法に大きく分類できる。

従来、スレッド分割の手法としては経験的手法が用いられてきた。しかし、経験的手法による分割では効率的なスレッドに分割できるとは限らない。

一方、遺伝的アルゴリズムを使う場合には、より効率的なスレッドに分割できる可能性がある。本稿では、遺伝的アルゴリズムを用いたスレッド分割法の提案と簡単なベンチマークプログラムによる評価を行う。

A Thread Generating Method with Genetic Algorithms

Kiyokazu Katayama, Shuichi Ichikawa, Toshio Shimada

School of Engineering, Nagoya University
Furo-cho,Chikusa-ku, Nagoya, 464-01, Japan

Abstract

It was suggested that processors should support multiple threads generated from one program and execute them at once. There are two main techniques in thread generating method, heuristic algorithms and genetic algorithms.

Heuristic algorithms has been used in thread generating method. However, effective threads aren't always generated in heuristic algorithms.

It is possible to generate more effective threads in Genetic algorithms. In this paper, we propose new thread generating method with genetic algorithms and evaluate it using benchmark programs.

1 はじめに

近年、プロセッサ内の演算実行部の使用率を向上させるために、マルチスレッドスーパスカラ方式の研究がなされている [1]。この方式は、プログラム中の並列実行可能部分 (スレッド) を、同時に実行することによって演算実行部の使用率を高めるものである。そのため、コンパイラには高度なスレッド抽出能力が要求される。

プログラムを複数のスレッドに分割する手法には、大きく分けると経験的手法を用いる方法と遺伝的アルゴリズムを用いる方法がある。従来、経験的手法によってプログラムを複数のクラスタに分割する研究がなされており [2, 3, 4]、これらをスレッド分割に用いることができる。しかし、経験的手法を用いた場合には、一つのスレッドの生成は、プログラム中の局所的な部分からのみ行われるため、効率的なスレッドに分割できるとは限らない。

一方、遺伝的アルゴリズムを用いた場合、スレッドの生成は乱数によってプログラム全体から行われるため、より効率的なスレッドに分割できる可能性がある。そこで本稿では、遺伝的アルゴリズムを用いたスレッド分割法の提案と簡単な評価を行う。

2 スレッド分割

スレッド分割とは、プログラムを複数の並列実行可能部分に分割することであり、一般的には、プログラムをデータフローグラフに変換し、そのグラフを分割することによってスレッドの生成が行われる。

ここで、図 1のような分割を考える。このような分割を行うと、実行時にデッドロックが起こり得る。従って、このようにスレッド間でリンクがループするような分割を行ってはならない。

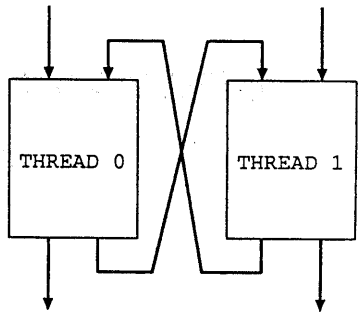

図 1: デッドロックが起こり得る分割

3 遺伝的アルゴリズムによるスレッド分割

遺伝的アルゴリズムは、組合せ問題に対して強力な探索能力を有しており、グラフ分割における遺伝的アルゴリズムの適用の研究がなされている [5, 6]。

データフローグラフを基にしたスレッド分割も、データフローグラフという有向グラフのグラフ分割と捉えることができる。そこで、スレッド分割に遺伝的アルゴリズムを適用する。

3.1 遺伝的アルゴリズム

遺伝的アルゴリズムは生物の進化過程を模式した確率的探索アルゴリズムである。遺伝的アルゴリズムでは、個体 (解候補) に交叉・突然変異・選択の 3 つの操作を繰り返し適用することにより、よりよい解の探索を行う。

交叉は、二つの個体間でその一部を交換し、新たに二つの個体を生成する操作である (図 2)。突然変異は、個体の一部を確率的に変化させる操作である (図 3)。選択は、評価値 (適応度) に応じて個体を確率的に選ぶ操作である。

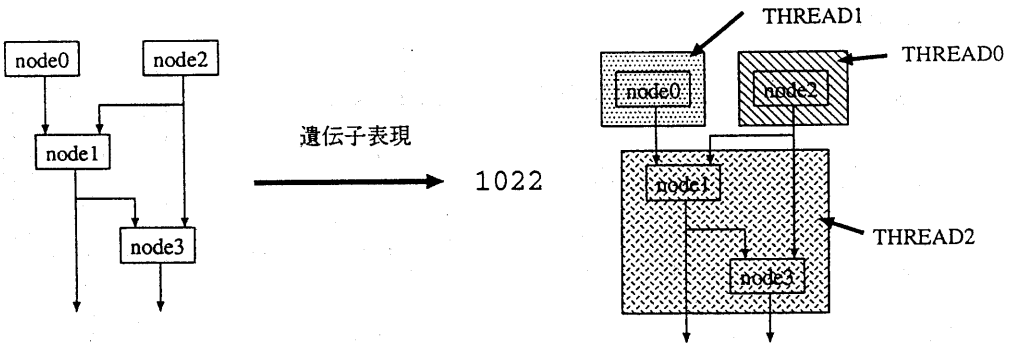

図 4: 遺伝子表現

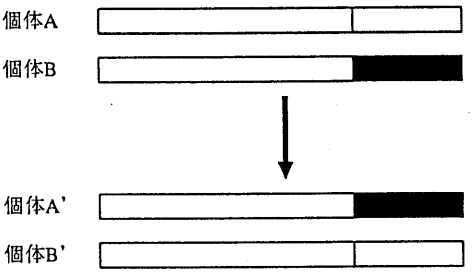

図 2: 交叉の実行

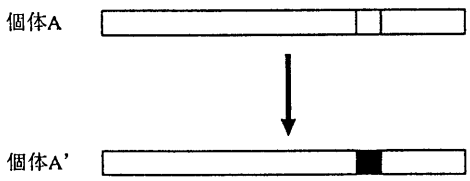

図 3: 突然変異の実行

3.1.1 遺伝子表現

遺伝的アルゴリズムを適用するためには、個体を遺伝子表現する必要がある。本研究では、ノード数が n 個のデータフローグラフを n ビットで表現する。各ビットは、データフローグラフのノードと一対一に対応しており、ビットの値によって、そのノードが属するスレッド番号を決定している (図 4)。

3.1.2 適応度関数

スレッド間で通信を行うことによってデータの送受を行うと、大きなコストが必要となる。そこで、スレッド分割を行う際にはスレッド間のリンクの数が少なくなるように分割を行う必要がある。また、各スレッドで実行される命令数はほぼ等しくなるように分割を行うと、実行時間が短くなる。そこで、適応度関数 $fitness$ を以下のように決定した。

$$fitness = \frac{link\_fitness + \sum_{thread} node\_fitness}{2}$$

$$link\_fitness = \exp(-0.1 \times link)$$

$$node\_fitness =$$

$$\frac{1}{1+\exp(-node+0.8\times all\_node/partition)}$$

$$-\frac{1}{\exp(-node+1.2\times all\_node/partition)}$$

link	:	スレッド間の全リンク数
node	:	各スレッドのノード数
all_node	:	全ノード数
partition	:	最大分割数

3.2 ビットシフト操作

グラフ分割問題に遺伝的アルゴリズムを適用すると、通常の突然変異は局所解から抜け出すためには十分とは言えない。そこで、適当な確率で個体の各ビットの値を反転する必要がある [5, 6]。

本手法では、個体のビットの値をシフトする事によって局所解に収束する事を防いでいる (図5)。これは、個体の各ビットの値が 0,1 の二値ではないためである。

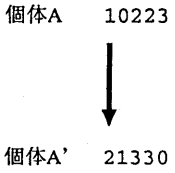

図 5: ビットシフト (4スレッド)

3.3 デッドロック回避

単純に遺伝的アルゴリズムを適用し、スレッドに分割しただけでは、スレッド間でループが生成されデッドロックが発生する可能性がある。そこで、デッドロックが発生しないことを保証するために、スレッド間のループを取り除く必要がある。本手法では、デッドロックの回避のために、ループが形成されている複数のスレッドを一つのスレッドにマージする手法を採っている (図6)。

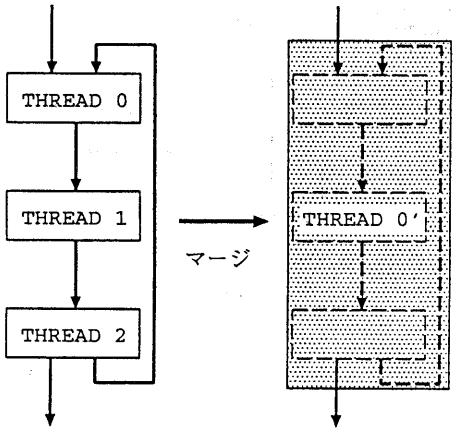

図 6: デッドロック回避法

4 評価

本手法の有効性を確かめるためにリバモアループのカーネル 1 から 12 に本手法によるスレッド分割を行った。遺伝的アルゴリズムにおいて、交叉には 1 点交叉を、選択にはスケーリングを行わない基本的な選択方法を採用した。なお、リバモアループは DFC II[9, 10] で書かれたものを手でデータフローグラフに変換し、そのデータフローグラフに対して本手法によるスレッド分割を行った。

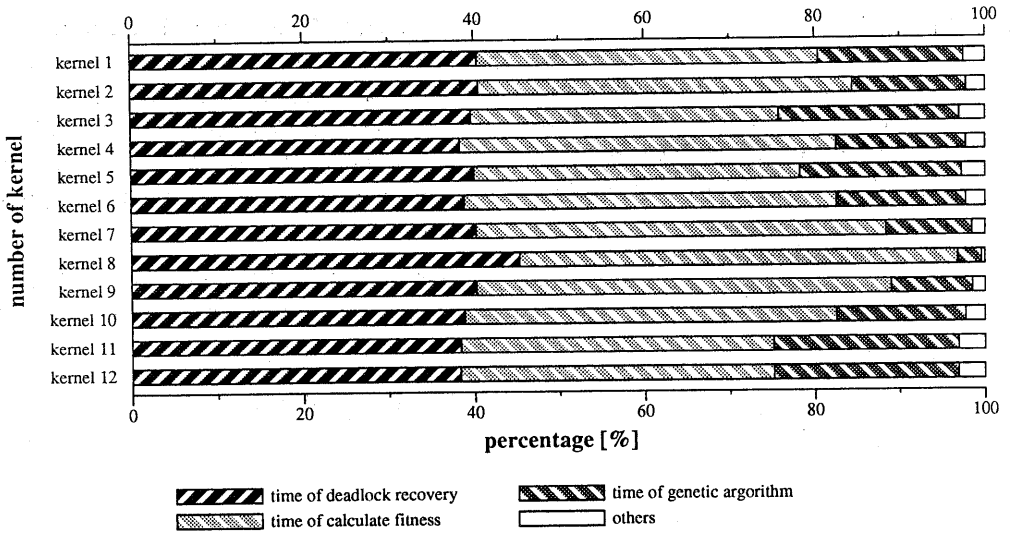

図 7: 各機能ブロックの実行割合

4.1 パラメータ

遺伝的アルゴリズムのパラメータには以下のものを用いた。

個体数	:	50
交叉生起確率	:	0.5
突然変異生起確率	:	0.7
ビットシフト生起確率	:	0.1
最大分割スレッド数	:	4

5 結果

表 1は、各カーネルをデータフローグラフ表現した時のノード数, リンク数及びスレッド分割に要した時間を示している。また、図 7は、実行時における各機能ブロックの全実行時間に占める割合を示している。

kernel	node	link	time[s]
1	23	29	8.04
2	33	55	4.39
3	15	21	1.57
4	29	44	7.63
5	15	25	5.29
6	27	40	6.90
7	15	67	8.57
8	107	97	124.30
9	19	77	9.76
10	57	87	21.10
11	51	19	3.49
12	55	19	5.69

表 1: ノード数とリンク数と実行時間

6 おわりに

本稿では、遺伝的アルゴリズムを用いたスレッド分割法を提案し、リバモアループによる簡単な評価を行った。その結果、一部のカーネルではスレッド分割に要する時間が非常に大きい事が明らかになった。

今後、以下の課題が残っている。

- 経験的手法との比較。
- より多くのプログラムでの評価。
- 経験的手法を取り入れた高速化。
- 生成されたスレッドの計算機上での評価。

参考文献

[1] Robert A. Iannucci, Guang R.Gao, Robert H.Halstead,JR., Burton Smith: "Multithreaded Computer Architecture:A Summary of The State of The Art", Kluwer Academic Publishers,1994.

[2] Rovert A. Iannucci:"Toward a Dataflow/Von Neumann Hybrid Architecture", In Proc. 15th Int. Symp. on Comp. Arch., pp.131-140, Hawaii, May, 1988

[3] Walid A. Najjar, Lucas Roh, A.P.Wim Bohm:"Initial Performance of a Bottom-Up Clustering Algorithm for Dataflow Graphs", Proc.of IFIP WG 10.3 Working Conference on Architecture and Compilation Techniques for Find and Medium Grain Parallelism(A-23), M.Cosnard,K.Ebcioglu and J-L.Gaudiot(eds.), Elsevier Science Publishers B.V.(North-Holland), pp.91-102,1993

[4] Culler, D. E., A. Sah, K. E. Schauser, T. von Eicken and J. Wawrzynek: "Fine-grain Parallelism with Minimal Hardware Support: A Compiler-Controlled Threaded Abstract Machine", 4th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems, pp.164-175, April 1991.

[5] 丸山, 小長谷, 小西: "遺伝的アルゴリズムを用いた並列グラフ分割アルゴリズム", 情処学論,Vol.34,No.4,pp.556-567,1993.

[6] 小長谷: "計算機が自分で問題を解決し始める夢", 第 15 回学際研究会議,No.29,pp.22-31,1995.

[7] 坂和正敏, 田中雅博: "遺伝的アルゴリズム", 朝倉書店,1995.

[8] 北野宏明 編: "遺伝的アルゴリズム", 産業図書,1993.

[9] 関口, 島田, 平木: "同期構造を埋め込んだ SIGMA-1 用高級言語 DFCII", 情処学論,Vol.30,No.12,pp.1639-1645,1989.

[10] 関口, 島田, 平木: "並列記述言語 DFCII の命令レベルデータ駆動計算機に対する構造文処理", 情処学論,Vol.30,No.10,pp.1454-1462,1990.