

並列実装による遺伝的アルゴリズムの耐故障性向上

北 咲也[†] 佐藤 裕二^{‡,‡}

法政大学大学院情報科学研究科[†] 法政大学情報科学部[‡]

1. はじめに

進化計算において、探索性能の向上と高速化を実現するために並列計算機上での実装に関する研究は以前から行われていた[1][2]。また、近年では一般に PC などにおいて使用されている CPU のマルチコア化や、GPU による汎目的計算 (GPGPU) の普及などが進んでいる。これにより一般の PC 上でも比較的容易に並列計算環境を構築し、実行することが可能になってきている。しかしながら、並列計算環境には従来の非並列計算環境と比較して、非常に多い回数の通信を並列環境中のシステム及びプロセッサで行うという特徴が存在する。この特徴のために、システム全体に耐障害性を与える事は並列計算環境にとって重要な課題のひとつである。何故ならば、システムの一部で発生した過渡的障害がシステム全体に波及するような耐障害性の低いシステムであれば、並列化されていないシステムに比べて遥かに信頼性の低下するためである。

本稿では、並列化による耐障害性の向上についてははじめに論じる。次に提案に基づいて並列化 GA を実装したプログラムを用いた評価実験を行い、最後に実験について考察及びまとめを行う。

2. 耐障害性と並列実装

耐障害性とは、システムの一部に障害が発生した場合においてもシステム全体としては正常に処理を続行することが可能な性質のことを言う。なお、本稿では特に過渡的障害、即ち物理的な破損等以外の処理及び通信上で起こりうる一時的な障害について論じる。一般にシステムを並列化して実装を行うと、システム上に存在する複数、もしくは全てのプロセッサが同時に処理を行うことになる。このことから、複数のプロセッサを持つようなハードウェアにおいては、ハードウェアの性能を並列化により充分に活用することが可能になる。本論文では上で述

べたような並列化における恩恵を受けつつも、並列化 GA において耐障害性を持たせるために各プロセッサ間の独立性の高い手法を提案する。

3. 並列化 GA

並列化 GA では、手法により違いはあるが、探索中の遺伝的操作はプロセッサごとに独立に行なっている場合が多くみられる。一方、並列化されたシステムにおいてプロセッサ間でのデータのやり取りや通信を頻繁に行う場合、1つのプロセッサで発生した過渡的障害が他のプロセッサにも波及し、システムが正常に動作できなくなる恐れがある。即ち、耐障害性を持たない可能性が生じるのである。この例として、あるプロセッサにおいてプログラム上のエラーにより生じた不正なデータを元に他のプロセッサが更に不正なデータを生成してしまい、最終的にシステム全体で誤った処理を実行してしまうという事象が想定される。したがって、探索においてプロセッサ間の独立性が高い手法の場合、その手法自体がある程度の耐障害性を持つことが予想される。このことから、並列化 GA と耐障害性は比較的親和性が高いといえることができる。本稿では Fig.1 に示したように、各プロセッサが独立に探索を行い、いずれか1つの探索が最適解に至れば探索を終了するという並列 GA のモデルを提案する。

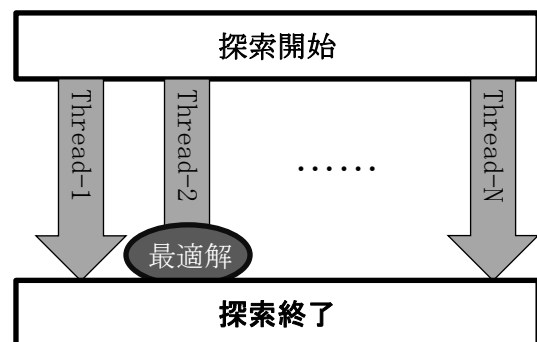


Fig. 1 提案手法のモデル図

Increase Fault Tolerance in GA by Parallel Processing

[†]Sakuya Kita (Graduated School of Computer and Information Sciences, Hosei University)

[‡]Yuji Sato (Faculty of Computer and Information Sciences, Hosei University)

4. 評価実験

4.1 評価方法

並列化による耐障害性を検証するために、提案モデルに基づいて実装を行ったプログラムを用いて評価実験を行う。なお、各プロセッサが一定の確率で本来の評価値の代わりに個体を取りうる範囲でランダムな値を評価値として返すようなコードを実験プログラム中に追加している。これは、プロセッサが個体の評価計算を正確に行えなくなるという過渡的故障を擬似的に再現するためである。

探索の精度と要する世代数を比較するために、アイテム数 40 のナップザック問題と De Jong の F2 関数最適化問題を障害発生率とスレッド数を変えながら、並列化された GA によって探索を行う。また、手法による耐障害性の差についても評価するために島モデル方式の並列化 GA でも実験を行う。

実験は、以下の環境上で設定されたパラメータに基づいて GA による探索を行うプログラムを実行して行う。個体数：100、選択手法：トーナメント選択(サイズ：4)、交叉率：0.7、突然変異率：0.01、実行回数：各 100 回

4.2 実験結果

表 1 にアイテム数 40 のナップザック問題を探索するのに要した平均世代数、表 2 に同じく De Jong の F2 関数最適化問題における平均世代数を示す。なお、最適解の探索自体はどの組み合わせにおいても 100 回中 100 回成功した。

Table 1. ナップザック問題における障害発生率及びスレッド数と探索に要した世代数の関係

Error Rate	1-Th.	2-Th.	4-Th.	8-Th.
0	12603	6249	3722	1934
0.00001	13626	7159	3825	1964
0.00005	13515	6791	3806	1949
0.0001	13465	7072	4007	2055
0.0005	14581	7344	4033	2067
0.001	14584	7773	4365	2007
0.005	15053	7414	4409	2078

Table 2. De Jong F2 関数最適化問題における障害発生率及びスレッド数と探索に要した世代数の関係

Error Rate	1-Th.	2-Th.	4-Th.	8-Th.
0	18287	10182	3925	399
0.00001	21139	11871	4059	512
0.00005	22125	12781	4635	509
0.0001	22379	12189	5024	593
0.0005	22860	13822	5478	616
0.001	23657	13104	7940	756
0.005	22917	15964	5760	1019

4.3 実験考察

表 1 及び表 2 から、まずスレッドの本数、即ち並列度が高いほど最適解を得るために要する世代数が少なくなっている事がわかる。また、ナップザック問題においては並列度が高くなるほど擬似エラー頻度に世代数が影響されにくくなっている。このことから、並列化された GA は並列化されていない GA に比べて一定の耐障害性を得ていると考えられる。また、関数最適化問題においても並列度が高まるに従って要する世代数は減少しているが、エラー頻度の増加に伴う世代数の増加率は増加傾向にある。この事象については要する世代数自体の少なさ、パラメータや GA 自体の設計、あるいは並列化により得られる耐障害性に問題依存性の存在する可能性などが原因として考えられる。今後、前述した事項の考察及び、より実用的な問題や GPGPU 等他の実行環境を用いて、並列化されたシステムにおける GA の優位性を既存研究との比較等を通して更に検証していくことが必要と考えられる。

5. まとめ

今回提案した「各プロセッサが独立に探索を行い、いずれか 1 つの探索が最適解に至れば探索を終了する」というモデルに基づく並列化による性能比較実験を通して、GA を並列実装することによりある程度の耐障害性が獲得されることが確かめられた。しかしながら DeJong F2 関数最適化問題においては、世代数の増加率について並列度と必ずしも逆比例する結果にはならなかった。この事象については前章で述べたような原因が考えられるが、更なる検討及び調査が求められると言える。

謝辞

本研究の一部は北海道大学情報基盤センター 2011 年度公募型共同研究および 2011 年度法政大学サスティナビリティ研究教育機構の助成を受けた。

参考文献

- [1] 佐藤裕二ほか「GPU による進化計算の並列化とサスティナビリティに関して」進化計算学会シンポジウム 2011 年, S4-06, (2011)
- [2] 藤村雄一ほか「マルチプロセッサによる遺伝的アルゴリズム処理のフォールトトレランス実験」電子情報通信学会総合大会講演論文集 1995 年, 情報・システム(1), 281, (1995)
- [3] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning" Kluwer Academic Publishers, Boston, MA (1989)
- [4] J. H. Holland "Adaptation in Natural and Artificial Systems" University of Michigan Press, Ann Arbor (1975)