Vol.2019-SE-203 No.7
2019/12/12

PfGA を用いたテストスイート自動生成機構

高橋 太郎[1]　高田 眞吾[1]

概要：ソフトウェアテストにおいて，様々な条件を網羅したテストスイートを用意する必要がある．しかし，そのようなテストスイートを手動で作成することは労力がかかってしまう．そのため，テストスイートを自動生成する手法が多数提案されている．その中で遺伝的アルゴリズムを利用した研究があるが，そこではパラメータ設定が重要であり，パラメータの値によって結果が大幅に変わる．本研究では，パラメータフリー遺伝的アルゴリズム（PfGA）を用いたテストスイート自動生成手法を提案する．PfGA とは遺伝的アルゴリズムの一種であり，事前にパラメータを決定する必要がない手法である．この手法を用いて，パラメータを決定するために必要な労力を割くことなく，高いカバレッジのテストスイートを生成する．

キーワード：ソフトウェアテスト，遺伝的アルゴリズム，パラメータフリー

1. はじめに

ソフトウェアは日常生活を便利にするためにいたるところで活用されている．様々な分野においてユーザは，ソフトウェアに対してより高度で，より多様な機能を求めている．そのため，ソフトウェア開発者は質の高いソフトウェアを開発することが求められている．開発したソフトウェアが正しく動作するか，不具合を起こさないかを確認する工程をソフトウェアテストと呼ぶ．ソフトウェアテストは，質の良いソフトウェアを開発するために必要不可欠な工程となっている．

ソフトウェアテストを実行するにあたって，カバレッジ (網羅率) が高いテストスイート (テストケースの集合) を作成することが重要となる．しかし，ソフトウェアが複雑になるとテストスイートに求められる条件も増大してしまうため，手動でテストスイートを作成することは非常に労力のかかる作業である．多くのテストを実行しなければならない場合も多く，それに伴ってコストが増大してしまう．

そのため，テストスイートを自動生成する手法の研究が行われている．その中でもヒューリスティックサーチ技法をソフトウェアテストの分野に適用させた Search-Based Software Testing の研究が盛んであり，特に遺伝的アルゴリズムが注目されている．Fraser ら [1] は，遺伝的アルゴリズムを利用してテストスイートを自動生成する EvoSuite を提案した．しかし，遺伝的アルゴリズムの欠点として，パラメータをあらかじめ決定しなければならない点が挙げられる．

遺伝的アルゴリズムのパラメータとは，主に初期個体数，交叉率，突然変異率のことを指す．パラメータの値によって結果が大きく異なってしまうが，その選定には困難が伴う．最適なパラメータ設定に関する効果的かつ確実な理論は存在しないことが研究により証明されており [2]，何度も実験を繰り返すことでしか適切なパラメータを見つけることはできない．

本稿では，以上の問題点に対処するため，パラメータフリー遺伝的アルゴリズム（PfGA）を用いたテストスイート自動生成手法を提案する． PfGA [3] とは，事前にパラメータを設定する必要のない遺伝的アルゴリズムであり，パラメータの設定に労力をかけることなく一定の効果を発揮することができると考えられる．

以降，2 章では本研究の基盤技術である遺伝的アルゴリズム，PfGA について述べる． 3 章では関連研究について述べる． 4 章では本研究で提案する機構とその詳細について述べる． 5 章では評価について述べる．そして 6 章では結論と今後の課題を述べる．

2. 基盤技術

本研究では，テストスイートの自動生成に遺伝的アルゴリズムの考え方を用いる．本章では，一般的な遺伝的アルゴリズム，そして遺伝的アルゴリズムを改良した PfGA について述べる．

[1] 慶應義塾大学
Keio University

2.1 遺伝的アルゴリズム

遺伝的アルゴリズムとは，生物の適応進化に関する自然淘汰をモデル化し，生物の進化を模したアルゴリズムである．あらかじめ初期個体数，交叉率，突然変異率，個体選択方法などを設定する必要があり，これらをパラメータと呼ぶ．遺伝的アルゴリズムは高い広域探索能力を持つが，一方で局所探索解に陥ってしまう可能性があるという問題点も存在する．

以下に遺伝的アルゴリズムの具体的な手順を述べる．

(1) あらかじめ設定された初期個体数の個体をランダムに生成し，初期個体群として設定する．
(2) 各個体のそれぞれの適応度を計算する．
(3) あらかじめ設定された選択方法によって個体の選択を行う．この際，適応度が高いほど選択されやすいような工夫が行われている．
(4) 設定された交叉率によって交叉を行い，新しい個体を生成する．
(5) 設定された突然変異率によって突然変異を行い，新しい個体を生成する．
(6) 終了条件を満たしている場合，その時点での最良の個体を最適解として終了する．
終了条件を満たしていない場合，終了条件を満たすまで (2) から (5) を繰り返す．

2.1.1 遺伝的アルゴリズムに必要なパラメータ

遺伝的アルゴリズムにはパラメータが必要であり，この値が結果を大きく左右する．以下にパラメータの詳細を述べる．

(1) 初期個体数 : アルゴリズムの最初に用意する個体の数．多すぎると解が収束しにくく，少なすぎると局所最適解に陥りやすくなる．
(2) 交叉率 : 交叉が起こる確率．高いほど交叉が起きやすくなる．
(3) 突然変異率 : 突然変異が起こる確率．高いほど突然変異が起きやすくなる．
(4) 個体選択方法 : 交叉，突然変異を起こす個体を選択する方法．
EvoSuite で採用されているランキング選択の他に，適応度に比例して選択確率を決定するルーレット選択，個体群から一定数の個体をランダムに取り，その中で最も適応度の優れているものを次世代に残すトーナメント選択等が存在する．

2.2 PfGA

パラメータフリー遺伝的アルゴリズム (PfGA) とは，遺伝的アルゴリズムを改良した手法である．この手法は，1998 年に Sawai ら [3] が提案したものである．遺伝的アルゴリズムではパラメータを設定しなければならないが，パラメータの良し悪しで結果が変わってしまうという問題点がある．この問題点を解決するため，パラメータを設定する必要のない遺伝的アルゴリズムである PfGA が考案された．

PfGA の特徴として，乱数発生器さえ設定すればどのようなものに対しても使用できるという点である．実行対象や実行目的によらず一様に使用することができるため，設計負荷を軽減することができる．また，通常の遺伝的アルゴリズムと比較して局所最適解に陥りにくい．

以下に PfGA の具体的な手順を述べる．

(1) 2 個の個体をランダムに生成し，局所集団として設定する．
(2) 個体群からランダムに 2 個体を選択し，交叉させる．この 2 個体を親個体とし，交叉させてできた 2 個体を子個体とする．
(3) 子個体の一方に突然変異を起こす． 2 個体のうちどちらに突然変異を起こすかは，等確率にランダムで決定する．
(4) 親個体との適応度を計算し，以下の操作を行う．
  (a) 子 2 個体がともに親 2 個体より適応度が優れている場合，子 2 個体及び適応度の優れている方の親個体，合計 3 個体を局所集団に戻す．
  その際局所集団の個体数は 1 増加する．
  (b) 親 2 個体がともに子 2 個体より適応度が優れている場合，親 2 個体のうち適応度が優れている方のみを局所集団に戻す．
  その際局所集団の個体数は 1 減少する．
  (c) 親 2 個体のうちどちらか一方のみが子 2 個体より適応度が優れている場合，親 2 個体のうち適応度が優れている方の個体及び子 2 個体のうち適応度が優れている方の個体，合計 2 個体を局所集団に戻す．
  (d) 子 2 個体のうちどちらか一方のみが親 2 個体より適応度が優れている場合，子 2 個体のうち適応度が優れている方の個体のみを局所集団に戻し，新たにランダムな個体を生成して局所集団に追加する．
(5) 局所集団の個体が 1 個になった場合，ランダムな個体を生成して局所集団に追加する．
(6) 各個体のそれぞれの適応度を計算する．
(7) 終了条件を満たしている場合，その時点での最良の個体を最適解として終了する．
終了条件を満たしていない場合，終了条件を満たすまで (2) から (6) を繰り返す．

3. 関連研究

本章ではテストスイートの自動生成に関わる関連研究，その中でも Search-Based Software Testing, そして本研究に用いる EvoSuite について述べる．

3.1 Search-Based Software Testing

Search-Based Software Testing とは，ヒューリスティック探索アルゴリズムを用いてテストスイートを生成する技術の総称である．ヒューリスティック探索アルゴリズムとは，問題に関する情報を活用し，より優れた解を導き出すアルゴリズムを指す．解が優れているかの判断は，達成したい要件に対する達成度合いを定量的に評価する評価関数によって行われる．評価関数は重視する内容によって異なり，コードカバレッジ，実行時間等に関する関数が存在する．コードカバレッジよりもミュータントの方がバグとの相関が高いという研究結果も報告されているものの [4]，ミュータントテスト時間がかかりすぎてしまうという欠点がある．ヒューリスティック探索アルゴリズムで使われるアルゴリズムとしては遺伝的アルゴリズム，メタヒューリスティクスである山登り法や焼きなまし法等を中心に研究されている．

McMinn[5] によると，この分野の研究は 2000 年頃から盛んになったといわれている．Xue-ying ら [6] は，テストスイートを削減するため回帰テストに遺伝的アルゴリズムを適用した．Bo ら [7] は，タブーサーチと遺伝的アルゴリズムを組み合わせてテストケースを生成する手法を提案した． Waeselynck ら [8] は焼きなまし法をソフトウェアテストの分野に適用した．Mahmood ら [9] は，Android アプリに対するテストシナリオを生成する EvoDroid を開発した．

3.2 EvoSuite

Fraser ら [1] は， EvoSuite という Search-Based Software Testing を実装したテストスイート自動生成ツールを開発した．また，後に Pablo ら [10] の提案した遺伝的アルゴリズムと局所探索を組み合わせた Memetic Algorithm という探索手法を EvoSuite に応用することに成功している [11].

EvoSuite の流れを**図 1** に示す．EvoSuite は Java 言語で書かれたクラスに対してテストスイートを自動生成するツールである．EvoSuite は個体をテストスイートとし，遺伝子をテストケースとする遺伝的アルゴリズムを用いている．具体的には初めにあらかじめ設定された個数のテストスイートをランダムに生成し，または終了条件を満たすまで選択・交叉・突然変異の操作を繰り返す．また，EvoSuite はテストスイート生成が終了した後にテストスイートのサイズを最小化する操作も同時に行う．

3.2.1 選択

EvoSuite では交叉を行う親個体などの選択方法としてランキング選択を採用している．ランキング選択とは各個体を適応度に従ってランク付けし，ランクごとに決められた確率で選択する方法である．

3.2.2 交叉

交叉対象の 2 個体 A,B それぞれを前半部分と後半部分に分解する．その後，A の前半部分と B の後半部分を引き継いだ個体，B の前半部分と A の後半部分を引き継いだ個体を生成する．

3.2.3 突然変異

EvoSuite の突然変異には，テストスイートに対する突然変異と，テストケースに対する突然変異の 2 種類がある．

テストスイートに対する突然変異は，テストスイートに対してランダムなテストケースを挿入する操作である．突然変異率を x とすると，まず x の確率でランダムなテストケースを挿入する．挿入された場合，次に x^2 の確率でさらにランダムなテストケースを挿入する．これをテストケースが挿入されなくなる x^n まで繰り返す．

テストケースに対する突然変異は，突然変異前の個体に元から備わっていたテストケースの内の 1 つに対して，文を挿入，削除，変更する操作である．この 3 種類がそれぞれ設定された確率で行われるため，全ての操作が実行される場合もある．

3.2.4 エリート保存

EvoSuite では，最も優れた個体をそのまま次世代に引き継ぐエリート保存を採用している．これにより，適応度の優れた個体を淘汰することがなくなるため，どのタイミングで終了しても常にの最良解を出力出来る状態を保つことができる．

4. 提案手法

本稿では， PfGA を適用したテストスイート自動生成機構を提案する．実装は EvoSuite をベースに遺伝的アルゴリズムの部分を PfGA で実現する．この手法を用いることにより，テスト対象の特徴に左右されることなく，一定以上の成果が得られると考えられる．

4.1 提案手法の流れ

提案手法の流れを**図 2** に示す．テスト対象のクラスファイルを入力とし，終了条件を満たした時に最も結果の優れているテストスイートを解として出力する機構となっている．

4.1.1 局所集団の個体数

最初に生成される個体は 2 個体であり，以降の交叉などの操作の結果次第で局所集団の個体は増減する． 2 個体未満になった場合はランダムに生成された個体を局所集団に加えることとし，2 個体未満にならないようにする．

局所集団の個体数が一定でないことによってより優れた探索を行うことが可能になる．局所集団の個体数が多い場合は，現在の局所集団に優れている個体があるためその個体付近を優先して探索するため，解の収束を進めることができる．逆に局所集団の個体数が少ない場合は，新しいラ

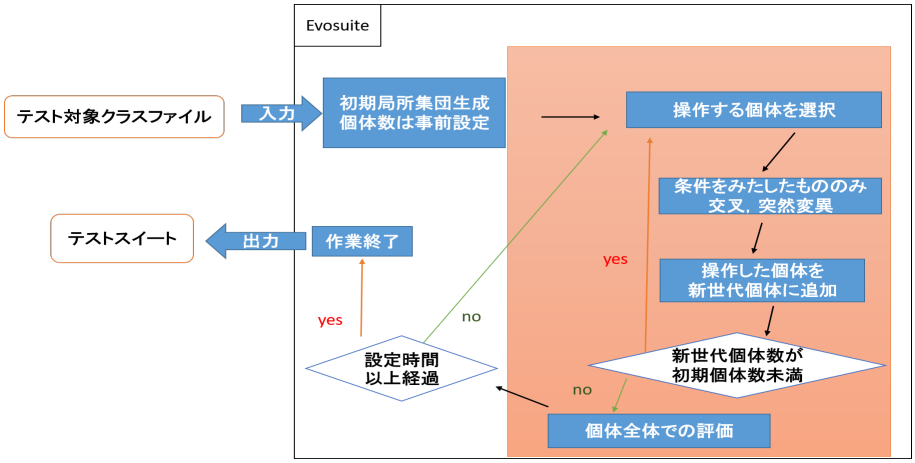

図 1　通常の EvoSuite の流れ

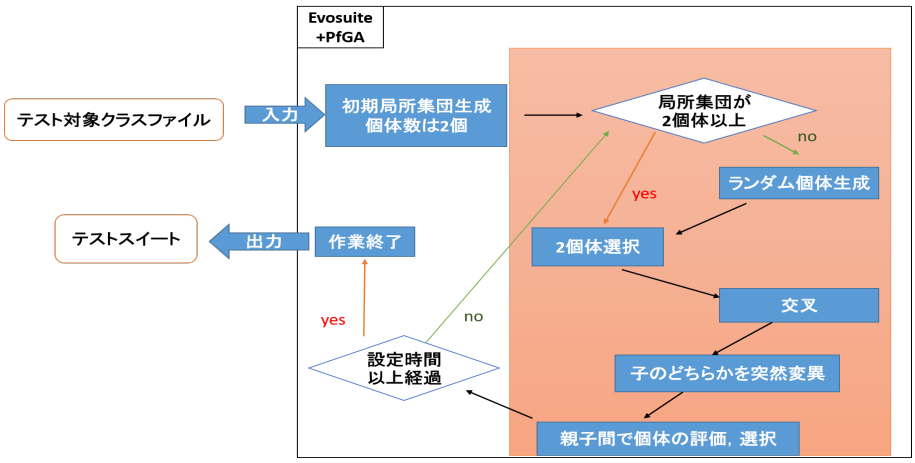

図 2　PfGA を用いた EvoSuite の流れ

ンダムな個体を探索することになるため局所最適解に陥りにくい．

4.1.2　個体の選択

PfGA の説明でも記述した通り，交叉，突然変異させる個体の選択方法は完全にランダムである．これは，個体の選択方法によっても結果が変わってしまうことと，同じ選択方法でも数値の設定により変化が起こることが理由である．

4.1.3　交叉

PfGA のアルゴリズムにおいて，交叉方法としては多点交叉が一般的である．多点交叉とは，ランダムな個数の交叉点を選び，その交叉点の前後で遺伝子を入れ替える方法である．これに対し，EvoSuite で従来行われていた交叉方法は一点交叉という．

本研究では PfGA の考え方に従い，テストスイート同士を多点交叉させた．ただし，テストスイートごとにテストケースの含まれている数が異なるため，テストケースの少ない方の個体以下の範囲内の数の交叉点を生成することにした．

4.1.4　突然変異

EvoSuite の突然変異は，テストスイートに対してのものとテストケースに対してのものの 2 種類ある．本研究では，交叉で得られた子個体の内 1 個体に対して両方の突然変異を行う．

テストスイートに対する突然変異については，従来の EvoSuite では新たに挿入するテストケースの数も一定ではなかった．そのため，挿入するテストケースの数をパラメータに影響されないようにするためには突然変異の仕組みを変更する必要がある．そこで，本研究ではいかなる場合でも，テストスイートに対する突然変異を 1 度だけ行うことにした．

テストケースに対する突然変異については，従来の EvoSuite では挿入，削除，変更の突然変異が独立しており，複数の種類の突然変異が同時に行われる可能性があった．本研究では，必ず全ての突然変異を行う必要はないと考え，いずれか 1 種類の突然変異のみを必ず行うことにした．突然変異の種類の決定方法は，突然変異を行う度に等確率にランダムで決定する．

4.1.5 終了条件

あらかじめ時間を設定しておき，その時間内で得られた最後の世代の個体の中で最も優れている個体を解として出力する．時間は 60 秒に設定した．

また，終了時間に至る前にコードカバレッジが 100 %の個体が生成された場合，その時点でその個体を解として出力する．

5. 評価

本稿では，提案手法の評価について述べる．

5.1 Research Questions

本研究では以下の項目を RQ に定めた．

(1) RQ1：パラメータの違いによるカバレッジへの影響はどの程度出るのか?
遺伝的アルゴリズムはパラメータによって結果が異なるのか，また異なる場合はどの程度の差異が出るかを調査する．また，優秀なテストスイートを生成するパラメータの傾向を考察する．

(2) RQ2：個々のベストパラメータの値にばらつきがあるか?
複数のテストクラスにおいてそれぞれのベストパラメータを調査し，パラメータ設定の有用性を検証する．また，それらが全体の結果とどう関係しているかも調査する．

(3) RQ3：PfGA の利用は結果にどのような影響を与えるのか?
PfGA を用いることによって従来の EvoSuite と比較してよいテストスイートが得られるのかを調査し，EvoSuite の有用性を検証する．

5.2 実験方法

本実験の方法について以下に述べる．

5.2.1 実験手順

(1) 従来の遺伝的アルゴリズムに関して，複数のパラメータを用意し， EvoSuite に適用する．

(2) (1) で用意した機構と PfGA を適用した EvoSuite を用いて，それぞれ同じテストプログラムに対してテストスイートを自動生成する．

(3) (2) を各パラメータの組み合わせごとに 10 回ずつ行い，プログラム中の各クラスについてカバレッジの平均を求める．

(4) (3) で算出した値から，テストプログラムに含まれるパッケージごとに含まれるクラスのカバレッジの平均を算出する．また，全パッケージに含まれるクラスのカバレッジの平均も算出する．

5.2.2 パラメータについて

従来の遺伝的アルゴリズムにおけるパラメータの重要性を調査するために，パラメータによる結果の違いを調べることとした．**表 1** にパラメータの組み合わせに関して示す．各パラメータの実験値を組み合わせ，各パッケージごとに最も優秀な結果を残したものをベストパラメータとして記録した．なお，表 1 に記述したデフォルト値とは， EvoSuite にもともと設定されている値である．これは Fraser らが EvoSuite を開発する際実際に実験して導き出した値である．

表 1 パラメータの組み合わせ

パラメータ	実験値	デフォルト値
初期個体数	10 , 50 , 100	50
交叉率	0.45 , 0.60 , 0.75 , 0.90	0.75
テストスイートに対する突然変異率	0.1 , 0.01	0.1
テストケースに対する突然変異率	1/3 , 1/10	1/3
個体選択方法	ランキング選択，トーナメント選択	ランキング選択

表 2 評価で用いたプログラムに含まれるパッケージ

パッケージ	内容	総クラス数	総ブランチ数
ciphers	暗号化	6	2828
Conversion	変換	17	2619
DataStructures	データ構造	50	9542
divideconquer	分割	2	1239
DynamicProgramming	動的計画法	14	3177
Maths	計算	1	47
MinimizingLateness	遅延の最小化	1	112
Others	その他	40	5659
Search	探索	7	1323
Sort	並び替え	16	4346
合計		154	30892

5.2.3 実験に使用したプログラム

実験には，合計 157 個のクラス， 11 個のパッケージを持つ Java プログラム [12] をテスト対象のプログラムとして選択し，そのうちの 154 クラス，10 パッケージを使用した．**表 2** に Java に含まれるパッケージごとの詳細を示す．

5.3 結果

本実験で得られたベストパラメータを**表 3** に示す．表 3 は，各パッケージごとのベストパラメータと，プログラム全体としてのベストパラメータの組み合わせを示している．

本実験のカバレッジの結果を**表 4** に示す．表 4 は，実験対象の各パッケージにおけるカバレッジの平均を，PfGA を用いたとき，各パッケージのベストパラメータで EvoSuite を用いたとき，全てのパラメータ値の組み合わせで EvoSuite を用いたときの平均を示す．

表 3　ベストパラメータ

パッケージ	初期個体	交叉率 交叉率	テストスイート	テストケース	個体選択
ciphers	50	0.90	0.1	1/10	ランキング
Conversion	100	0.90	0.01	1/3	ランキング
Data Structures	50	0.75	0.1	1/3	ランキング
divideconquer	50	0.45	0.01	1/10	ランキング
Dynamic Programming	100	0.60	0.01	1/3	ランキング
Maths	50	0.75	0.1	1/3	ランキング
Minimizing Lateness	50	0.90	0.1	1/3	ランキング
Others	100	0.60	0.01	1/3	ランキング
Search	100	0.45	0.1	1/10	ランキング
Sort	50	0.60	0.1	1/3	ランキング
全パッケージ	50	0.75	0.1	1/3	ランキング

表 4　実験結果

パッケージ	PfGA	ベストパラメータ	全パラメータの平均値
ciphers	91.22 %	92.95 %	90.44 %
Conversion	95.94 %	96.08 %	92.71 %
DataStructures	89.76 %	89.84 %	88.03 %
divideconquer	85.35 %	90.61 %	83.92 %
Dynamic Programming	89.49 %	89.78 %	87.97 %
Maths	95.83 %	95.83 %	95.83 %
Minimizing Lateness	65.69 %	84.39 %	81.15 %
Others	91.36 %	91.40 %	90.33 %
Search	90.65 %	94.90 %	89.58 %
Sort	95.16 %	95.16 %	95.08 %
全パッケージ	91.31 %	90.98 %	89.99 %

5.3.1　RQ1：パラメータの違いによるカバレッジへの影響はどの程度出るのか?

表 3 の最後の行「XXX」にあるパラメータ値の組み合わせは EvoSuite のデフォルトパラメータと同じである．つまり，EvoSuite が提供しているデフォルトパラメータ値はどのようなクラスに対しても比較的良い結果を出せることを示している．また本実験では，全体の結果は個々のパッケージの結果と比較してあまり差がつかない結果となった．各パラメータについて静的に検証した結果，以下が見られた．

(1) 初期個体数： 50 個に設定した場合に優秀な結果を得やすく，次いで 10 個の場合が優秀であった．これらの結果より，一般的には個体数は多すぎると解の探索が進みにくいことがわかる．

(2) 交叉率： 確実な傾向を見つけることはできなかったが，交叉率が 0.45 の場合はテストスイートのカバレッジが低くなりやすかった．

(3) 突然変異率：突然変異率のみを変化させても結果があまり変わらず，傾向を見つけることはできなかった．

(4) 個体選択方法：ほとんどの場合で，ランキング選択の方が優秀な結果を得ることが出来た．特に初期個体数が多いとこの傾向は顕著であった．

5.3.2　RQ2：個々のベストパラメータの値にばらつきがあるか?

表 3 より，プログラム全体としてのベストパラメータはデフォルト値と同じ組み合わせだが，パッケージごとのベストパラメータはバラバラであった．デフォルト値は個別のベストパラメータにはなっていないことが多く，またプログラム全体のカバレッジが低いパラメータ値の組み合わせであっても特定のクラスに対してのみ優秀な結果を残す場合もあった．このため全てに対して最適であるパラメータを導き出すことは難しく，パラメータ設定の重要性が示された．

5.3.3　RQ3：PfGA の利用は結果にどのような影響を与えるのか?

表 4 より，プログラム全体としての結果を見ると，PfGA を用いた際のテストスイートは最も優秀であるということが示された．これは PfGA がパラメータによらない故に汎用性があるということだと考えられる．ただし，個別のパッケージで見た場合には最良の結果を出しているという訳ではなかった．また，一部のパッケージ，一部のクラスに対しては平均を下回る結果となってしまっている．よって PfGA はどのようなクラスに対してもある程度優秀な結果を出すことができるが，必ずしも最適な解を見つけ出せるわけではないということが示された．

6.　結論

本論文では，EvoSuite に対して PfGA を用いたテストスイート自動生成機構を提案した．パラメータの設定が不要であるこの手法を利用することにより，どのようなプログラムに対しても一定以上の性能を誇るテストスイートを生成することができると考えられる．

今後の課題としては，より優れた結果を効率よく生成するための PfGA の改良，ソフトウェアテストの幅を広げるためのコードカバレッジ以外の評価関数への対応等が挙げられる．また，より多くのクラスファイルで比較を行い，PfGA の有用性をより検証する必要がある．

参考文献

[1] Gordon Fraser, Andrea Arcuri. EvoSuite: automatic test suite generation for object-oriented software., *the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp.416-419, 2011

[2] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch,

Catherine Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning., *SCIENCE*, Volume 220, No. 4598, pp.671-680, 1983

[3] Hidehumi Sawai, Sachi Kizu. Parameter-Free Genetic Algorithm Inspired by "Disparity Theory of Evolution ", *Parallel Problem Solving from Nature, PPSN 1998*, pp.702-711, 1998

[4] Rene Just, Darioush Jalali, Laura Inozemtseva, Michael D. Ernst, Reid Holmes, Gordon Fraser. Are mutants a valid substitute for real faults in software testing?, *FSE 2014 Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp.654-665 , 2014

[5] Phil McMinn. Search-Based Software Testing: Past, Present and Future., *ICST Workshops 2011*, pp.153-163, 2011

[6] Xue-ying Ma, Bin-kui Sheng Cheng-qing Ye. Test-Suite Reduction Using Genetic Algorithm, *International Workshop on Advanced Parallel Processing Technologies* , pp.253-262, 2005

[7] Bo Yu, Yemei Qin. Generating test case for algebraic specification based on Tabu search and genetic algorithm, *Cluster Computing*, Volume 20, Issue 1, pp.277-289, 2017

[8] Helene Waeselynck, Pascale Thvenod-Fosse, Olfa Abdellatif-Kaddour. Simulated annealing applied to test generation: landscape characterization and stopping criteria., *Empirical Software Engineering*, Volume 12, Issue 1, pp35-63, 2007

[9] Riyadh Mahmood, Nariman Mirzaei, Sam Malek. EvoDroid: segmented evolutionary testing of Android apps, *FSE 2014 Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* , pp.599-609, 2014

[10] Pablo Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts - Towards Memetic Algorithms, *Caltech Concurrent Computation Program* , 1989

[11] Andrea Arcuri, Gordon Fraser, and Phil McMinn. A Memetic Algorithm for Whole Test Suite Generation., *Journal of Systems and Software,* Volume 103, pp.311-327, 2015

[12] GitHub-TheAlgorithms/Java, *https://github.com/TheAlgorithms/Java*(last accessed Nov 14, 2019)