3L-8

ミューテーション法と遺伝的アルゴリズムを用いたテストデータの自動生成

末広暁久† 佐々木亮太† 芳賀博英†

同志社大学大学院工学研究科†

１． はじめに

本報告では，ソフトウェアテストにおける，テストデータの自動生成について述べる．具体的には，テストセットの評価法の一つであるミューテーション法と遺伝的アルゴリズムを用いて，ランダムに生成したテストデータを精錬して，所望の網羅基準を満たすデータを生成する方法を提案する．

２． 提案手法

２．１ ミューテーション法

ミューテーション法とは，ソースコードに故意に変更を加えたプログラム（ミュータント）を作り，それをテストセットが検出できるかどうかによって，テストセットのバグ検出能力を評価する手法である．テストセットのベンチマークとも言える．ミュータントの種類はいろいろあるが，今回用いたミュータントは表 2 の 3 種類である．

表 1: 生成するミュータントの種類

識別子	内容
COR	条件分岐の判定条件全体を true・false に置換する
CSR	条件分岐の判定条件内の，論理演算子で結ばれた各要素をそれぞれ true・false に置換する
BVI	条件分岐の判定条件内の，関係演算子で結ばれた数値型の各要素それぞれの左辺と右辺の値を+1 する

これらのミュータント群をテストデータ群が検出できたかどうか（元のプログラムとミュータントに同じテストデータを入力して違う出力が帰ってきたら検出できたと見なす）を摘出行列として記録する．

	T1	T2	T3	T4	T5	T6	T7	T8	T9
M1	1	1	1	1	1	1	1	1	1
M2	0	0	0	0	1	0	1	1	0
M3	1	0	1	1	1	0	1	0	0
M4	0	1	1	0	0	1	1	0	0
M5	0	0	0	0	0	0	0	0	1

図 1: 摘出行列の例

Study on the Automatic Testdata Generation with Mutation Analysis and Genetic Algorithm
†Akihisa Suehiro, Ryota Sasaki, Hirohide Haga,
Graduate School of Engineering, Doshisha University

２．２ テストデータ網羅基準

自動生成されたテストデータが，COR と CSR によって作られたミュータントを全て検出できた場合，テストデータはブランチカバレージ 100%を満たす．また，BVI によって作られたミュータントを全て検出できた場合，境界値カバレージ 100%を満たすことになる．

２．３ 遺伝的アルゴリズム（GA）

下の式を用いて，ミュータントを検出されにくさ（*strength*）で重み付けした総和をとるようにして，個々のテストデータの品質(*quality*)を定量化する．M はミュータント総数，T はテストデータ総数，DT_m はミュータント m を検出できたテストデータの総数，$detected_{mt}$ は摘出行列の m行 t 列の値である．

$$quality_t = \sum_{m=1}^{M} (detected_{mt} \cdot strength_m)$$

$$strength_m = log_{DT_m+1}(T+1)$$

そして品質の高いテストデータ同士の各構成要素を選択して交叉させることで，新しいテストデータを生成する．

２．３ 提案手法概要

提案手法を実装したシステム構成が図 1，実行の流れが表 1 である．

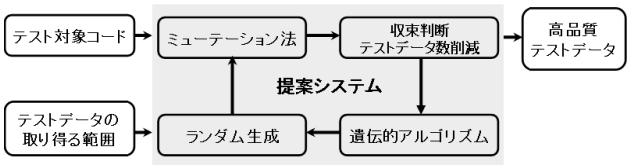

図 2: システム構成

表 2: 実行の流れ

Step1:	テストデータの取りうる範囲内でランダムにテストデータを生成
Step2:	ミューテーション法で摘出行列を作成
Step3:	テストデータ数の削減
Step4:	ミューテーションスコアが収束したら終了
Step5:	遺伝的アルゴリズムでテストデータ生成
Step6:	step1 に戻る

システムへの入力は C 言語のソースコードと，テストデータを構成する各パラメータの取り得る範囲であり，出力は，テストデータである．

まずテストデータの取り得る範囲内でランダムにテストデータを生成する(Step1)．次に，テスト対象コードのミュータントを作り，ミュータントとオリジナルのプログラムとでテストデータを実行し，実行結果とミューテーションスコア（検出率）を記録する(Step2)．その後，ミュータントの検出能力が包含関係にあるテストデータを削除する[1](Step3)．そして，ミューテーションスコアが前回のサイクルと同じ（新たに生成されたテストデータで検出できるようになったミュータントがない）場合は，十分な品質のテストデータが生成されたと見なして生成プロセスを終了する(Step4)．そうでない場合は，遺伝的アルゴリズムを用いて新たにテストデータを生成する(Step5)．そして Step1 に戻ってさらにランダム生成したテストデータを加えて，上記のサイクルを繰り返す(Step6)．

３．　評価実験

３．１　提案システムによるテストデータ生成

実装したシステムの評価実験として，月齢算出プログラム（年月日を入力してその日の月齢を求めるプログラム）のテストデータを自動生成した．生成サイクルに GA を含めた場合と，ランダムのみの場合の両方を行った．また，ランダム生成では結果にばらつきがあるため複数回実行する．結果は表 3 のようになった．

ミュータントは合計 169 個生成されたが，どのようなテストデータでも検出することのできないミュータントである等価ミュータントが 2 つ生成されたため，実験対象プログラムのミューテーションスコアの理論上最大値は 0.988 であった．また，実行にかかった時間は平均で 137 秒であった．

表 3: 提案システムによるテストデータ生成結果

	ランダム	GA
1	0.940	0.988
2	0.988	0.988
3	0.988	0.988
4	0.988	0.988
5	0.988	0.988
6	0.905	0.988
7	0.952	0.988
8	0.988	0.905
9	0.940	0.988
10	0.988	0.988
平均	0.966	0.979

３．２　人間によるテストデータ生成

情報工学を専攻している大学生・大学院生 3 人が，同じプログラムのテストデータを手動で作成した結果は表 4 のようになった．

表 4: 人間によるテストデータ生成結果

実験人	ミューテーションスコア	時間 (秒)
A	0.538	1280
B	0.532	1610
C	0.532	930
平均	0.534	1273

４．　考察

ランダム生成のみでは 10 回中 5 回しかブランチ・境界値カバレージが 100%にならなかったが，GA を用いることで，10 回中 9 回 100%になった．

提案システムは人間より 10 倍近く高速に，正確なテストデータを生成することができた．人間の作ったテストデータのカバレージが低い理由としては，逐次実行の後の方にある条件分岐の場合，そこに到達するまでに変数の内容などが書き換えられるため，逆算してテストデータを作る必要があり，そのときに間違いを犯しやすいことが考えられる．このように，単純なプログラムでも手作業で完全なカバレージを満たすことは難しいが，アルゴリズムを使うことによって，所望のテストデータを生成できた．

５．　まとめ

プログラムの制御フローを網羅するテストデータを非常に高い精度で自動的に生成することができた．この成果によって，定型的な作業を機械に任せることができ，人間はより高度な作業に専念できるだろう．

現在，テストデータとして適用できるデータ構造は，テスト対象関数の引数であり，数値型である場合に限られているため，今後，テストデータとして適用できるデータ構造の拡張による汎用化が必要であるだろう．また，この手法を応用して，組み合わせを考慮した基準である複数条件網羅などの，より高度な網羅基準を満たすテストデータの自動生成も可能になってゆく．

参考文献

[1] Paul Ammann, Jeff Offutt, "Introduction to Software Testing", Cambridge University Press, pp. 170-212, (2008)

[2] 斉藤孝志, 大久保弘崇, 粕谷英人, 山本晋一郎 "ミューテーション法を用いたテストセット構成支援に関する研究", 情報処理学会研究報告(2005)