情報処理学会論文誌 Vol.59 No.4 1215–1224 (Apr. 2018)

不具合組合せ特定のための機械学習による自動分類法の提案と評価

西浦 生成[1,2,a)] 崔 銀惠[2,b)] 水野 修[3,c)]

受付日 2017年8月2日, 採録日 2018年1月15日

概要：不具合組合せ特定とは，組合せテストの各テストケースの実行結果の成否から，バグを含むと思われるパラメータ値の組合せを特定する問題である．本研究では，機械学習を用いて不具合組合せを自動分類するための手法を提案し，その評価を行う．提案手法では，まず，組合せテストケースに含まれるパラメータ値の組合せとテスト結果の成否を学習モデルとしたロジスティック回帰分析を行い，それによって得られる回帰係数値から，各パラメータ値の組合せが不具合組合せである疑わしさを決定する．次に，各パラメータ値の組合せの疑わしさから，その組合せが不具合組合せであるか否かを自動分類するために，閾値決定法および最大距離分割法の 2 つのクラスタリング手法を適用する．最後に，実際にバグを含むオープンソースプロジェクトのプログラム flex，grep，make のテストスイートに対して提案手法を適用した比較評価実験を行うことで，提案手法の有効性を示す．

キーワード：組合せテスト，不具合組合せ特定，機械学習，ロジスティック回帰分析

Machine Learning Based Classification for Faulty Interaction Localization

Kinari Nishiura[1,2,a)] Eun-Hye Choi[2,b)] Osamu Mizuno[3,c)]

Received: August 2, 2017, Accepted: January 15, 2018

Abstract: Faulty interaction localization is a problem to identify interactions of parameter values causing failures, called faulty interactions, from given the results of executing combinatorial test cases. In this paper, we propose an approach using machine learning to automatically classify each combination of parameter values in test cases into either of a faulty interaction or not. In the proposed approach, we first determine the suspiciousness of each combination of parameter values to be a faulty interaction by using a logistic regression analysis whose training dataset is obtained from whether the combination is included in a test case or not and whether the test case is failed or not. In order to classify each combination of parameter values with the obtained suspiciousness into a faulty interaction or not, we next present the two methods: (1) a determination using a boundary value, and (2) a division using the maximum distance. Finally, we show the effectiveness of the proposed approach by conducting comparative evaluation experiments that apply the proposed methods to combinatorial test suites for multiple versions of three open source projects, flex, grep, and make.

Keywords: combinatorial testing, faulty interaction localization, machine learning, logistic regression

1. はじめに

ソフトウェアを開発する際に，リリース前にバグを発見するためにテストを行うことは，ソフトウェアの品質や信頼性という面で重要である．組合せテスト [7], [10] はブ

1 京都工芸繊維大学大学院工芸科学研究科情報工学専攻
Graduate School of Science and Technology, Kyoto Institute of Technology, Kyoto 606–8585, Japan
2 産業技術総合研究所情報技術研究部門
Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology, Ikeda, Osaka 563–8577, Japan
3 京都工芸繊維大学情報工学・人間科学系
Faculty of Information and Human Sciences, Kyoto Institute of Technology, Kyoto 606–8585, Japan

a) k-nishiura@aist.go.jp
b) e.choi@aist.go.jp
c) o-mizuno@kit.ac.jp

ラックボックステストの一種で，複数個のパラメータ値の組合せによって起こる不具合を見つけ出すことを目的とする．また，組合せテストによって不具合を検出した際，テスト結果からバグを含むと思われるパラメータの組合せ（Faulty Interaction，以下 FI と呼ぶ）を特定することを「不具合組合せ特定」と呼ぶ．

テストの実行結果の成否から不具合組合せを特定する手法の 1 つに，テスト成否から不具合組合せを容易に特定可能なテスト（LDA [2]，ELA [8] など）を事前に作成する方法があるが，テスト設計・実行のコストが高い．これに対し，与えられたテスト結果を分析して FI の特定を導く，より実用的な手法が考えられている [5], [13], [14]．

我々は先行研究 [11] において，後者の不具合特定手法として，組合せテストとその結果からロジスティック回帰分析を用いて，ある組合せが FI である可能性を定量的に計算する手法を提案した．ロジスティック回帰分析は機械学習の一種であるロジスティック回帰を用いた分析手法であり，0 と 1 に漸近するロジスティック関数を回帰曲線として回帰を行うことで得られる各説明変数の重み（回帰係数値）を，目的変数の持つ増減への寄与度として，数値的に分析することを目的としている．我々の提案する手法では，t 個のパラメータからなる FI を特定したい場合，組合せテストに含まれる大きさ t の組合せすべてを抽出して説明変数とし，各テストの成否を目的変数とした学習モデルを構築する．そのモデルを用いてロジスティック回帰分析を実行することで，各説明変数の回帰係数値という形で分析値が得られる．これによって各組合せの存在がテスト失敗に与える影響の大きさ，すなわち，各組合せが FI である可能性を数値として得ることができる．

本論文では，さらに，各組合せに対して得られた分析値の分布から FI を決定するための 2 つの自動分類法を提案し，それらを用いたときの不具合組合せ特定の精度を比較評価する．1 つ目は，固定的な閾値を設けてそれ以上の分析値を持つ組合せを FI とする方法（M1），2 つ目は，正の分析値を持つ組合せのうち，分析値間の最大距離を持つ区間を境界として 2 つに分割し上位のものを FI とする方法（M2）である．

提案手法の適用評価のために，プログラムとテストを含むオープンなソフトウェア関連成果物リポジトリである SIR（Software-artifact Infrastructure Repository）[4] から 3 つのプロジェクト「flex」「grep」「make」のテストデータに対して実験を行った．実験に使用するテストスイートは，リポジトリにあらかじめ存在する全組合せ網羅の実行済みテストスイートに加え，それぞれのシステムモデルと制約仕様を組合せテスト生成ツール pricot [1] へ入力して生成した，2 項目間の組合せを網羅するペアワイズテストスイートの 2 種類を用いた．それぞれのテストスイートにおける各テストケースの成否は，SIR から取得したテスト履歴から得られる．

提案手法では，t 個のパラメータからなる FI を特定したい場合，まず，組合せテストに含まれる大きさ t の組合せ（以降，t-tuple と呼ぶ）をすべて抽出する．次に，各 t-tuple の疑わしさ（すなわち，FI である可能性）をロジスティック回帰分析で計算する．最後に，提案する自動決定手法を用いて FI であるか否かを決定する．適用実験においては，flex，grep，make の合計 42 個のバージョンを対象に，実際に求められるべき FI（以降，real-FI と呼ぶ）と，我々の提案する 2 つの FI 決定法によって求められた FI を比較し，提案手法の精度評価を行った．実験の結果，提案手法が非常に高い精度で FI を特定できることを示す．

以降の論文の構成は次のとおりである．2 章では，本研究の対象である組合せテストと不具合特定，本研究で用いるロジスティック回帰分析について説明する．3 章では，提案する不具合組合せ特定法のプロセスについて説明し，4 章では，提案手法を適用した評価実験の実施について，5 章では，実験結果とその考察について述べる．6 章で関連研究について述べ，最後に，7 章でまとめと今後の課題を述べる．

2. 準備

2.1 組合せテストと不具合特定

組合せテスト [7], [10] は，システムのパラメータが取る値の組合せに注目して，その組合せによって引き起こされる不具合を検出するテスト手法である．組合せテストのシステムモデル（*System Under Test*，*SUT*）は，パラメータ，パラメータ値，制約式の 3 つ組から構成される．たとえば，**表 1** のシステムモデルは，3 つのパラメータ P_1, P_2, P_3 を持ち，各パラメータは 2 つまたは 3 つの値を持つ．また，$(P_3 = v_6) \rightarrow (P_2 \neq v_3)$ という制約を持ち，$P_3 = v_6$ と $P_2 = v_3$ の組合せは許されないことを表す．

テストケースは，システムモデルの各パラメータへの値の割り当てで，制約を満たすものであり，テストスイートはテストケースの集合である．たとえば，**表 3** の tc_1=(v_1, v_3, v_5) はテストケースであり，$\mathcal{T}_1$ は 7 つのテストケースの集合からなるテストスイートである．

すべての組合せを網羅するテスト（全組合せテスト）は，

表 1 例題システムモデル（System Under Test）
Table 1 An example system under test (SUT).

Parameter	*Values*
Debug mode (= P_1)	on (=v_1), off (=v_2)
Bypass use (= P_2)	on (=v_3), off (=v_4)
Fast scanner (= P_3)	FastScan (=v_5), FullScan (=v_6), off (=v_7)
Constraint : (Fast scanner = FullScan) → (Bypass use ≠ on)	

表 2 パラメータ値のペア
Table 2 Parameter value pairs.

Param. pairs	*Parameter-value pairs*
(P_1, P_2)	(v_1, v_3), (v_1, v_4), (v_2, v_3), (v_2, v_4)
(P_1, P_3)	(v_1, v_5), (v_1, v_6), (v_1, v_7), (v_2, v_5), (v_2, v_6), (v_2, v_7)
(P_2, P_3)	(v_3, v_5), (v_3, v_7), (v_4, v_5), (v_4, v_6), (v_4, v_7)

表 3 ペアワイズテスト（$\mathcal{T}_1$）
Table 3 A pair-wise test ($\mathcal{T}_1$).

tc	P_1	P_2	P_3
1	v_1	v_3	v_5
2	v_2	v_4	v_5
3	v_1	v_4	v_6
4	v_1	v_3	v_7
5	v_2	v_4	v_6
6	v_2	v_4	v_7
7	v_2	v_3	v_7

表 4 全組合せテスト（$\mathcal{T}_2$）
Table 4 An all-pair test ($\mathcal{T}_2$).

tc	P_1	P_2	P_3
1	v_1	v_3	v_5
2	v_1	v_3	v_7
3	v_1	v_4	v_5
4	v_1	v_4	v_6
5	v_1	v_4	v_7
6	v_2	v_3	v_5
7	v_2	v_3	v_7
8	v_2	v_4	v_5
9	v_2	v_4	v_6
10	v_2	v_4	v_7

理想的であるが，そのサイズはシステムサイズに対して指数的に増加するため，現実には難しい場合が多い．そのため，ある組合せ数 t を設定し，t 個のパラメータ間でとりうる値の組合せ（以降，t-tuple と呼ぶ）をすべて網羅するテスト（t-wise テスト，または，t-way テスト）が実用的とされている．$t = 2$ の場合の t-way テストは最も広く用いられており，「ペアワイズテスト」と呼ばれる．**表 2** は，表 1 の例題システムモデルにおける制約を満たすすべてのパラメータ値のペアであり，また表 3 の $\mathcal{T}_1$ は，これらのペアすべてを網羅するペアワイズテストの例である．一方，**表 4** の $\mathcal{T}_2$ は，例題システムモデルの 3-way テスト，かつ，全組合せテストである．

組合せテストにおいて，ある t-tuple が不具合を引き起こす場合，すなわち，FI（Faulty Interaction）である場合，その t-tuple を含むテストケースの実行結果は失敗（fail）となる．t-way テストはすべての t-tuple を網羅しているため，組合せ数 t までの FI が存在する場合はテスト実行結果にかならず fail が存在し，不具合があることを検出可能である．そのようなテスト実行結果の成否から，実際にどの t-tuple が FI であるかを特定する問題が「不具合組合せ特定」問題である．不具合組合せは 1 つと仮定する研究もあるが，我々の研究ではより実用的に不具合組合せは複数存在しうると仮定する．

たとえば，**表 5** は，1-tuple の (v_3) と 2-tuple の (v_2,v_6) が不具合を引き起こす場合の例題ペアワイズテストに対するテスト実行結果である．この場合，1-tuple で FI のものは (v_3) の 1 つであり，2-tuple で FI のものは (v_2,v_6)，および，(v_3) を含む 2-tuple である (v_1,v_3)，(v_2,v_3)，(v_3,v_5)，(v_3,v_7) の 5 つである．また，ペアワイズテスト結果から，(v_1,v_5)，(v_1,v_7) は失敗したテストケースのみに含まれており，FI に分類される．たとえば，全組合せテストを使用する場合は，テストケース $tc3$=(v_1,v_4,v_5) $\in \mathcal{T}_2$ の実行結果が成功となり，(v_1,v_5) は FI に分類されない．(v_1,v_7) も同様である．このように，使用するテストスイートによって FI 特定の能力は異なる．

表 5 ペアワイズテスト $\mathcal{T}_1$ とテスト結果の例
Table 5 Example test results for $\mathcal{T}_1$.

tc	P_1	P_2	P_3	実行結果
1	v_1	v_3	v_5	fail
2	v_2	v_4	v_5	pass
3	v_1	v_4	v_6	pass
4	v_1	v_3	v_7	fail
5	v_2	v_4	v_6	fail
6	v_2	v_4	v_7	pass
7	v_2	v_3	v_7	fail

2.2 ロジスティック回帰分析

本研究ではロジスティック回帰分析を用いて，テストスイートを構成する各テストケースに含まれる任意のタプルの疑わしさ（FI である可能性）を算出する．ロジスティック回帰分析 [3] はロジスティック回帰モデルを利用した要素分析の手法であり，目的変数が 0 か 1 の二値をとる場合に有効である．ロジスティック回帰モデルは一般化線形モデルの一種であり，通常の線形回帰モデルでは直線に回帰させるのに対してロジスティック回帰モデルでは 0 と 1 に漸近する（ロジスティック）シグモイド曲線に回帰させる．ロジスティック回帰モデルは以下の式で表される．

$$Pr(Y=1|X)=\frac{1}{1+\exp[-(\alpha+\beta_1 x_1+...+\beta_k x_k)]} \quad (1)$$

これは，ある目的変数 Y が 1 をとる確率が，回帰係数 β_k によって重みづけられた説明変数 x_k のとる値に依存することを表している．このとき，高い回帰係数を持つ説明変数ほど，その変位による目的変数への影響が大きい．ロジスティック回帰分析は，学習データから回帰曲線を学習することによって各説明変数の回帰係数を決定し，それぞれの説明変数が持つ目的変数への寄与度を数値的に得ることを目的としている．

3. 提案手法

この章では，組合せテストの不具合組合せ（FI）を推定する提案手法を説明する．提案手法の概要を**図 1** に示す．提案手法の入力は，(1) 各テストケースの成否を対応させたテストスイートと，(2) 求めたい FI のサイズ t である．提案手法は大きく次の 2 つのステップに分かれる．

表 6 表 5 の入力例で $t = 2$ の場合の学習データ
Table 6 Learning data for the example in Table 5 ($t = 2$).

	テスト	実行	学習データ															
tc	P_1 P_2 P_3	結果	$e_{(v_1,v_3)}$	$e_{(v_1,v_4)}$	$e_{(v_1,v_5)}$	$e_{(v_1,v_6)}$	$e_{(v_1,v_7)}$	$e_{(v_2,v_3)}$	$e_{(v_2,v_4)}$	$e_{(v_2,v_5)}$	$e_{(v_2,v_6)}$	$e_{(v_2,v_7)}$	$e_{(v_3,v_5)}$	$e_{(v_3,v_7)}$	$e_{(v_4,v_5)}$	$e_{(v_4,v_6)}$	$e_{(v_4,v_7)}$	R
1	v_1 v_3 v_5	fail	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1
2	v_2 v_4 v_5	pass	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0
3	v_1 v_4 v_6	pass	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
4	v_1 v_3 v_7	fail	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1
5	v_2 v_4 v_6	fail	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1
6	v_2 v_4 v_7	pass	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0
7	v_2 v_3 v_7	fail	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1

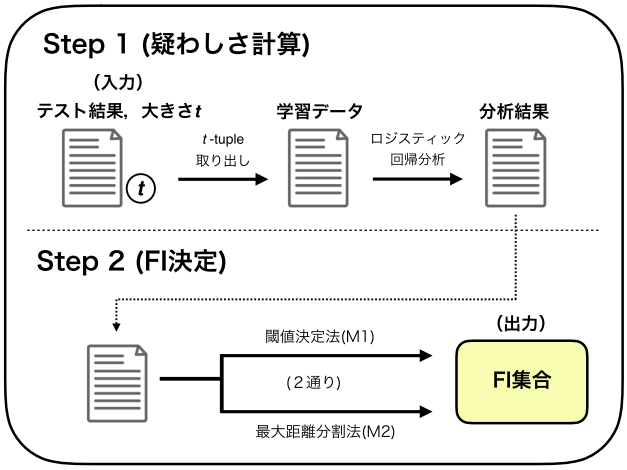


図 1 提案手法の概要
Fig. 1 The overview of the proposed approach.

Step 1 疑わしさ計算．テストスイートに含まれる各組合せ（t-tuple）の疑わしさ（FI である可能性）をロジスティック回帰分析によって計算する．

Step 2 FI 決定．得られた各 t-tuple の分析値を用いてその組合せが FI であるか否かを 2 つのクラスタリング手法—閾値決定法，最大距離分割法—のいずれかで決定する．

以降では，例題を用いながら，提案手法への入力と各手法の手順について説明する．

3.1 入力

3.1.1 テストスイートとテスト結果

本手法では，全テスト実行済みのテストスイートに，各テストの実行結果「成功（pass）か失敗（fail）か」を対応させたものを入力とする．表 5 は入力例に対応する．

ここで，すべてのテストケースの実行結果が pass であるテストスイートは，FI が存在しないため入力として適切ではない．同様に，すべてのテストケースの実行結果が fail であるテストスイートは FI の特定が不可能であるため適切ではない．

3.1.2 FI サイズの決定

本手法では，異なるサイズの FI を同時に求めることはできない．そのためテスト中の t 個のパラメータのタプル（以下，t-tuple）に FI があるかどうかを調べる操作を繰り返していく．基本的には，まず大きさ 1 の tuple を FI だと想定し，$t = 1$ から開始して，t の値を順に増やしながら繰り返していくことになる．単一パラメータでのテストがすでに完了している場合は $t = 2$ から開始するとよい．また特定のサイズの FI の存在を調べたい場合はその値を t に設定すればよい．

3.2 出力

提案手法によって FI であると判定された大きさ t のタプルの集合が出力として得られる．

3.3 Step 1：疑わしさ計算

3.3.1 存在表（学習データ）の作成

調べたい FI のサイズ t を決定したら，各テストケースが含む t-tuple とテスト成否の情報を表す存在表の作成を行う．この存在表がロジスティック回帰分析の入力として使用する学習データとなる．存在表の作成は次の 3 段階の手順で行う．(1) まず，テストスイート内に出現するすべての t-tuple を抽出する．(2) 次に，各テストケースに各 t-tuple i が含まれるか否かを表すブール変数 e_i を用意する．テストケースが t-tuple i を含む場合は $e_i = 1$，含まない場合は $e_i = 0$ である．(3) 最後に，各テストケースの実行が失敗したか否かを表すブール変数 R を用意する．実行が失敗した場合は $R = 1$，成功した場合は $R = 0$ である．

例として，表 5 と $t = 2$ を入力とした場合，2-tuple を抽出して作成した学習データを**表 6** に示す．この例では 2-tuple は全部で 15 種類存在する．

3.3.2 ロジスティック回帰分析の実行

t-tuple を説明変数，R を目的変数として，各 t-tuple i に対してロジスティック回帰分析を行い，回帰係数を分析値として計算する．

表 7 に入力例に対してロジスティック回帰分析を適用

表 7 各 2-tuple の分析値一覧，および，提案手法 M1（閾値決定法），M2（最大距離分割法）による FI 分類の結果例
Table 7 Analysis values for 2-tuples and FI classification results by M1 and M2.

2-tuple	回帰係数	M1	M2	real-FI
(v_3,v_7)	22.56	FI	FI	FI
(v_1,v_5)	22.56	FI	FI	FI
(v_3,v_5)	22.56	FI	FI	FI
(v_1,v_6)	22.56	FI	FI	FI
(v_1,v_7)	22.56	FI	FI	FI
(v_1,v_4)	22.56	FI	FI	FI
(v_2,v_3)	22.56	FI	FI	FI
(v_1,v_3)	22.56	FI	FI	FI
(v_2,v_7)	0.00	-	-	-
(v_4,v_6)	0.00	-	-	-
(v_2,v_4)	−0.69	-	-	-
(v_2,v_5)	−22.56	-	-	-
(v_4,v_7)	−22.56	-	-	-
(v_1,v_6)	−22.56	-	-	-
(v_1,v_4)	−22.56	-	-	-
(v_4,v_5)	−22.56	-	-	-

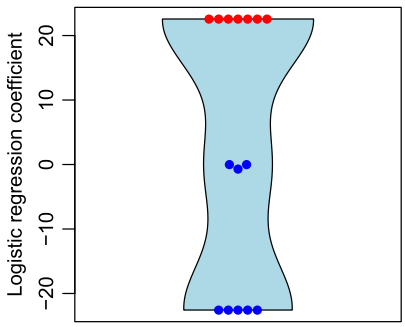


図 2 入力例に含まれる 2-tuple の分析値の分布
Fig. 2 The distribution of analysis values in Table 7.

することで得られた 2-tuple の分析値の一覧を示す．また，**図 2** に 2-tuple の分析値の分布を示す．図の赤点は実際に FI であるもの，青点は FI でないものを表す．また図の横軸は同一値の個数を表し，水色の領域は分析値の密度分布を表す．この図では，求められた分析値の分布が 3 カ所にまとまっていることが確認できる．

3.4 Step 2：FI の自動決定

Step 1 では各タプルの持つ FI としての疑わしさの値を算出した．これらの値を利用して，たとえば値の高い順に何らかの操作を行うといった運用は可能であり，そうした利用を行う場合であればこの段階で操作を終了してもよい．ただし，「不具合組合せ特定」を自動的に FI を特定するプロセスとして考えた場合，得られた疑わしさの値から自動的に FI の集合を抽出し出力する方法を考案することは有用である．こうした自動抽出の方法として，Step 2 として以下に記す方法を提案する．図 2 でも見られるように，前工程で求められた分析値は，大きく次の 3 つに分類できる．

- 0 から離れた顕著な正の値を持つもの．
- 絶対値が 0 に近い，微細な値を持つもの．
- 0 から離れた顕著な負の値を持つもの．

分析値が高いほどその組合せが FI である可能性が高くなるので，このうち，本手法では 1 つめの「0 から離れた顕著な正の値を持つもの」を FI として自動分類したい．

自動分類の方法としては様々なものが考えられるが，我々は，次の 2 つの方法を提案し，以降の実験でその精度を比較評価する．

M1 閾値決定法：固定的な閾値を設けてそれ以上の分析値を持つものを FI とする．

M2 最大距離分割法：正の範囲で分析値間の距離が最大の区間を境界として 2 つに分割し，上位に属するものを FI とする．

それぞれの方法について以降の各項で説明する．

3.4.1 M1：閾値決定法

最も簡単な方法として，閾値を設けてそれ以上の分析値を持つ t-tuple を FI とする方法が考えられる．この閾値は，前述の「0 から離れた顕著な正の値のもの」と「0 付近の微細な値のもの」を分断することが期待されるが，それを果たす適切な閾値は SUT やテストスイートの性質に依存すると予想され，一意に決定することは難しい．手法の説明のため，ここでは閾値を 10.0 と設定するとする．10.0 は，表 7 において分布のほぼ中間に位置する値である．

たとえば，表 7 に示す各組合せの分析値一覧に対して，10.0 以上の回帰係数を持つタプルは 6 つあり，FI 分類結果は表 7 の M1 のようになる．結果として，7 つの 2-tuple：(v_3,v_7)，(v_1,v_5)，(v_3,v_5)，(v_2,v_6)，(v_2,v_3)，(v_1,v_7)，(v_1,v_3) を FI と決定する．実際の 7 個の FI（real-FI）に対して，すべての FI を正しく特定できた結果となる．

3.4.2 M2：最大距離分割法

提案手法 M2 では，次の操作によって FI 決定を行う．まず，全 t-tuple を分析値の降順でソートする．次に分析値が 0 以下のものを除外する．ここに分析値 0 のダミーデータを加える．ここから隣り合う分析値間の距離のうち，最大のものを求める．求めた最大距離によって二分割される集団のうち，分析値の大きいほうに属するタプルを FI とする．

ここで，分析値が 0 以下のものを除外するのは，FI とそれ以外の組合せとの分析値の境界を正の範囲に限定したいためである．また分析値 0 のダミーデータを加えるのは，データセットによっては「絶対値が 0 に近い，微細な係数を持つもの」が出現せず，正しい分割が期待できない場合があるためである．

表 7 の例題に提案手法 M2 を適用する過程を**表 8** に示す．分析値でソート済みのリストから 0 以下のものを除外し，ダミーデータを加えたものから，隣り合う距離を計算

表 8 最大距離分割法（M2）によるクラスタリング
Table 8 Clustering by M2.

2-tuple	分析値	距離	クラスタ
(v_3,v_7)	22.56	-	1
(v_1,v_5)	22.56	0	1
(v_3,v_5)	22.56	0	1
(v_2,v_6)	22.56	0	1
(v_2,v_3)	22.56	0	1
(v_1,v_7)	22.56	0	1
(v_1,v_3)	22.56	0	1
(v_4,v_6)	0.00	22.56	2
(v_2,v_7)	0.00	0	2
(dummy)	0	0	2

して最大のものを求めると，(v_1, v_3) と (v_4, v_6) の間の距離が得られる．この区間を境界としてタプルを二分割し，上位に属する 7 つのタプルを FI とする．

FI 分類結果は表 7 の M2 のようになる．結果として，M1 と同様，実際の 7 個の FI（real-FI）に対して，すべての FI を正しく特定できた結果になっている．

4. 評価実験

我々は Python で提案手法の実装を行い，評価実験を行った．実装において，ロジスティック回帰分析の実行には，Python の統計モデルパッケージ statsmodels の一般化線形モデルを扱う方法を用いた．実験は，Macbook Pro（Retina 13-inch，Late 2013），2.8 GHz Intel Core i7，16 GB 1,600 MHz DDR3 で Python 3.4.3 を用いて行った．

4.1 実験対象

本実験では，オープンソースプロジェクト「flex」「grep」「make」に対して作成されたテストスイートを用いる．これらのテストスイートとバグを含む複数バージョンのプログラムに対するテストスイートの実行結果を，公開されたソフトウェア関連成果物リポジトリである Software-artifact Infrastructure Repository（SIR）[*1]から入手できる．

適用実験は，これらのプロジェクトにおける全組合せテストスイートとペアワイズテストスイートの 2 種類のテストスイートの実行結果を対象として行う．全組合せテストスイートはすべての入力の組合せを網羅したテストスイートであり，ペアワイズテストスイートはある 2 つのパラメータのとりうる値の組合せを網羅したテストスイートである．

全組合せテストスイートは SIR から入手できる．SIR から入手した全組合せテストスイートには単一のパラメータ入力によるテストケースとすべてのパラメータを用いたテストケースが混在しているので，実験では単一のパラメータ入力によるテストケースを除外して使用する．

[*1] http://sir.unl.edu/

```
Parameters:
...
 Debug mode:    # -d
   Debug_on.
   Debug_off.

 Bypass use:    # -Cr
   Bypass_on.   [property Bypass]
   Bypass_off.

 Fast scanner:  # -f, -Cf
   FastScan.    [property FastScan]
   FullScan.    [if !Bypass][property FullScan]
   off.         [property f&Cfoff]
...
```

図 3 flex のテストプランの一部
Fig. 3 A part of the test plan for flex.

表 9 flex，grep，make のシステムモデル（SUT）サイズ
Table 9 SUT sizes for flex, grep, and make.

Proj.	Model size
flex	パラメータと値のサイズ 29; $3^{23}4^46^2$
	制約のサイズ 97; $2^{712}22^124^225^{17}26^9$
grep	パラメータと値のサイズ 14; $2^43^14^35^16^19^111^113^120^1$
	制約のサイズ 87; $2^{433}3^{27}4^87^516^124^127^128^131^{10}$
make	パラメータと値のサイズ 22; $2^23^124^45^26^17^1$
	制約のサイズ 79; $2^{526}21^122^123^124^325^726^9$

またペアワイズテストスイートは，SIR にある flex, grep, make のテストプランからシステムモデル（SUT）を作成し，そのシステムモデルからペアワイズテスト生成ツール「pricot」[1] を用いて作成した．たとえば，**図 3** は，SIR にある flex のテストプランの一部である．そこから作成したテストモデルの一部が，表 1 である．**表 9** は，各プロジェクトに対して作成したシステムモデルの大きさを表す．表で，パラメータと値のサイズ $k; g_1^{k_1} g_2^{k_2} \ldots g_n^{k_n}$ は，パラメータ数が k で，各 $i(1 \le i \le n)$ に対して g_i 個の値を持つ k_i 個のパラメータがあることを表す．制約のサイズは，制約式をパラメータに対する値割当てを表すブール変数の CNF 式 $l; l_1^{h_1} l_2^{h_2} \ldots l_m^{h_m}$ で表したとき，l 個のブール変数があり，各 $j(1 \le j \le m)$ に対して l_j 個のリテラルを持つ h_j 個の

節があることを表す．

用意した 2 種類のテストスイートを，flex，grep，make の総計 152 バージョンのプログラムに対して適用した結果，すべてのテストケースの実行が成功したバージョンとすべてのテストケースの実行が失敗したバージョンが出現した．これらのバージョンは組合せ不具合特定の対象として不適当であるため除外し，全組合せテストスイートとペアワイズテストスイートの両方に対して失敗テストケースを含む 42 バージョンを分析対象とした．

表 10，**表 11** に各プロジェクトのテストごとの使用バージョン数（#ver），テストケース数（#tc），含まれる 1-tuple および 2-tuple の総数（#t-tuple）を示す．また各バージョンに存在する real-FI の数（#real-FI）を「最小値–最大値（平均値）」の形式で示す．また本実験では，評価対象としてペアワイズテストスイートを使用する性質上，2-tuple 以下（1-tuple と 2-tuple）の FI の特定を行う．

M1 で使用する基準値の値は，手法説明時と同じく 10.0 に設定した．

表 10 全組合せテストの諸パラメータ
Table 10 Parameters for all-pair test.

			#t-tuple		#real-FI	
Proj.	#ver	#tc	t=1	t=2	t=1	t=2
flex	31	500	48	213	0-3 (1.2)	0-60 (21.4)
grep	9	440	43	433	0-25 (8.0)	0-370 (135.4)
make	2	768	32	179	1-3 (2.0)	18-51 (34.5)
All	42				0-25 (2.7)	0-370 (46.5)

表 11 ペアワイズテストの諸パラメータ
Table 11 Parameters for pair-wise test.

			#t-tuple		#real-FI	
Proj.	#ver	#tc	t=1	t=2	t=1	t=2
flex	31	27	48	213	0-3 (1.5)	1-90 (36.7)
grep	9	45	43	433	0-29 (9.7)	1-397 (158.0)
make	2	8	32	179	1-10 (5.5)	51-134 (92.5)
All	42				0-29 (3.4)	1-397 (65.36)

4.2 評価方法

本実験では，我々の手法が本来求めるべき FI（以下，real-FI）を特定できているのかを評価したい．提案手法により特定された FI と real-FI がどの程度合致しているかによって評価する指標としては，適合率（Prediction），再現率（Recall），および，正確度（Accuracy）を用いる．適合率は我々の提案手法が分類した FI のうちの，real-FI の割合を表す．適合率が 100%であれば，提案手法によって FI と分類されたものはすべて real-FI である．再現率は real-FI のうち，我々の提案手法によって FI と分類できたものの割合を表す．再現率が 100%であれば，提案手法によって FI と分類されたものはすべての real-FI を含んでいる．正確度はすべての t-tuple のうち，我々の提案手法によって FI であると正しく推定できた real-FI および FI ではないと正しく分類できた非 real-FI の合計の割合を表す．正確度が 100%であれば，提案手法は real-FI と非 real-FI を完全に分離できている．比較に用いる real-FI は，その組合せを含むテストケースがつねに失敗するものを虱潰し式に探索することによって求めた．また提案手法の実行速度を示すため，処理時間の計測も行った．

5. 実験結果と考察

5.1 実験結果

表 12 に，ペアワイズテストスイート（Pairwise）と全組合せテストスイート（All）を対象とした，提案手法による 1-tuple および 2-tuple の FI 特定の平均処理時間および平均精度を示す．また，総合的な平均処理時間と平均精度の値を下段の Avg. に記す．さらに，それぞれの評価指標で 100%の精度が出たバージョン数を#(100%) に記す．

5.2 考察

実験結果から，本手法はきわめて高い精度で real-FI を特定できていることが分かる．特に，FI 決定法 M1 では，適合率，再現率，正確度ともに，すべてのバージョンで 100%となっている．これは，実験に用いたどのようなパターンのテスト結果からも過不足なく real-FI を特定できたことを表す．FI 決定法 M2 についても，再現率はすべて

表 12 M1（閾値決定法），M2（最大距離分割法）の精度比較
Table 12 Accuracy of M1 and M2.

		time (s)			Precision (%)		Recall (%)		Accuracy (%)	
	t	Step 1	Step 2 (M1)	Step 2 (M2)	M1	M2	M1	M2	M1	M2
Pairwise	1	0.15	8.50×10^{-6}	2.24×10^{-5}	**100.00**	87.18	**100.00**	**100.00**	**100.00**	99.52
	2	1.96	60.95×10^{-6}	10.65×10^{-5}	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
All	1	0.18	8.30×10^{-6}	2.17×10^{-5}	**100.00**	73.68	**100.00**	**100.00**	**100.00**	99.90
	2	2.05	71.13×10^{-6}	11.01×10^{-5}	**100.00**	88.97	**100.00**	**100.00**	**100.00**	99.96
Avg.		1.08	3.72×10^{-5}	6.52×10^{-5}	**100.00**	89.00	**100.00**	**100.00**	**100.00**	99.62
#(100%)					**168**/168	143/168	**168**/168	**168**/168	**168**/168	143/168

のバージョンで 100%であり，適合率，正確度に関しても全 168 バージョン中 143 バージョンで 100%，適合率の平均値は 89.0%と，非常に高い精度で real-FI を特定できている．

M2 で適合率が 100%にならなかった 25 のバージョンについては，そのすべてのケースにおいて，特定対象としている組合せサイズの real-FI が存在しないことが原因となっている．これはたとえば，3-tuple の real-FI のみが存在し，2-tuple 以下の real-FI が存在しないとき，$t = 1$ および $t = 2$ を手法の入力として 1-tuple および 2-tuple の FI を特定しようとしても，実際に存在する 3-tuple の real-FI の特定には至らない．こうしたときに，分布間の距離に注目した M2 の方法では必ず 1 つ以上の組合せを FI と判定してしまうため，real-FI でない組合せを FI と判定してしまう．このことによって適合率および正確度が低下しているものと思われる．これは M2 の構造的欠陥である．またペアワイズテストを実験対象としたとき，失敗したテストケースが存在するならばそれは 2-tuple 以下の real-FI の作用である可能性が高い．本実験でペアワイズテストの 2-tuple を対象とした場合のみで M2 の適合率が 100%になっているのは，2-tuple 以下の real-FI が全バージョンで存在していることが原因である．また一方で，M1 では閾値を超える回帰係数を持つ組合せが 1 つもなければ「t-tuple の FI は存在しない」と正しい判定が可能であり，そのため本実験では M2 で起こるような誤判定もなく完全な特定が実現できている．

閾値を設定する M1 は今回すべてのバージョンで正確度が 100%となり，real-FI を完全に特定することができたが，これは今回選択した 10.0 という閾値が実験対象に対して適合した結果であり，閾値や実験対象を別のものに変えたときに同様な結果になる保証はない．適切な閾値を自動的に決定することは難しく今後の課題であり，現状では使用者が手動で適切な閾値を探り設定する必要があるため，自動で境界を作るがある条件下で精度の落ちる M2 とは一長一短である．これに関連して，**図 4** に実験対象に対して閾値を 0 から 30 まで 0.5 刻みで変更したときの M1 の平均正確度の変遷を示した．このグラフを見ると，閾値を 6.0 から 22.5 までの間に設定した場合に平均正確度は 1 となり，すべてのバージョンで M1 によって正しく real-FI を特定できることが分かる．もちろんこの結果を他の SUT ないしテストスイートにそのまま適用できるわけではないが，「完全な分断」を実現する閾値設定の幅はさほど狭くないことから，他の適用対象に対して手動で閾値を設定する場合にも適切な閾値を苦労なく見つけられる可能性は高いと予想できる．

また処理時間に関しては，総時間のうちロジスティック回帰分析で怪しさを求める Step 1 の処理が大部分を占める．Step 2 の M1 および M2 を比較すると M2 のほうが処理時間はわずかに長いが無視できる程度である．

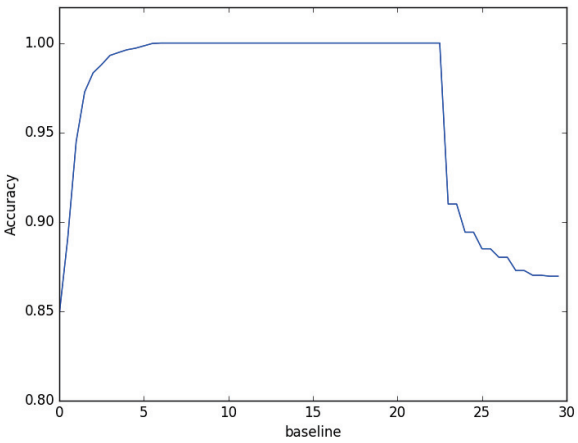


図 4 実験対象における閾値の設定と平均正確度の関係
Fig. 4 Relationship between the threshold and the average accuracy.

6. 関連研究

組合せテストの不具合組合せ特定に対するアプローチは，Adaptive アプローチと Non-Adaptive アプローチの大きく 2 つに分けられる．

Non-Adaptive アプローチには，テスト成否から不具合組合せを容易に特定可能なテストスイートを事前に設計する手法がある．LDA（Locating and Detecting Array）[2]，ELA（Error Locating Array）[8] などがその例である．たとえば，(d, t)-locating array は，d 個までのサイズ t の FI を特定可能なテストスイートである．Nagamoto らの研究 [9] では，与えられたペアワイズテストから Locating Array を生成する手法を提案する．Konishi らの研究 [6] では，最小サイズの (d, t)-locating array を SAT ソルバーを用いて生成する手法を提案する．このように LDA や ELA を設計するアプローチは，テスト結果から FI を容易に特定可能である一方，構築されるテストスイートは，テストの数が大きく，テスト設計・実行のコストが高い，という短所がある．また，Zhang ら [14] は，与えられた組合せテストの FI 探索を論理式の充足可能性（SAT）問題へ変換し，SAT ソルバーを用いて FI を探す方法を提案する．彼らの手法では 1 個の FI を見つけることを前提としており，複数の FI を前提している我々の研究とは異なる．

Adaptive アプローチでは，与えられた組合せテストとその実行結果から各タプルが FI である疑わしさを計算する手法を採用する．我々の提案手法はこちらに分類される．Yilmaz ら [13] は Classification Tree 手法を用いて，Ghandehari ら [5] は独自のヒューリスティック計算式に基づいて，FI の疑わしさを計算する手法を提案する．我々は，ロジスティック回帰を用いて疑わしさを計算し，さらに，FI であるか否かを自動分類するための 2 つの方法を提案し，比較評価した．関連研究とは実験に使用したデー

タセットが異なるが，Ghandehari らの論文によると，彼らの手法の適合率は 50–100%（平均 56%）とのデータがあり，提案手法の精度の良さを推測できる．また我々は，FI 決定の対象サイズを特定のサイズに限定せず一般化する手法 [12] も提案している．一方，FI の疑わしさ予測を基に Adaptive にテストケースを追加していく方法を先行研究 [5] では提案しており，我々の研究では今後の課題である．

7. 結論

本論文では，ロジスティック回帰分析を用いて組合せテストにおける不具合組合せの疑わしさを定量的に計算し，分析値の分布から自動的に FI を特定する 2 通りの手法を提案した．また，実際にバグを含むプロジェクトの組合せテストとその実行結果を対象に提案手法の 2 つの FI 自動分類法の比較評価実験を行い，提案手法の総合的な精度の高さを確かめるとともに，それぞれの方法の持つ特徴を示した．

今後の課題として，提案した 2 通りの FI 自動特定法の抱える問題点，すなわち M1 では適切な閾値を自動的に求められないこと，また M2 では求めるべき real-FI が存在しない場合にも FI を検出してしまう場合があることを解決する必要がある．またより多様な実験対象で評価実験を行うことや他の FI 特定法とのパフォーマンス比較を行うことなどがあげられる．

謝辞 この研究の一部は日本学術振興会科学研究費補助金 16K12415 の助成を受けて実施された．

参考文献

[1] Choi, E., Kitamura, T., Artho, C., Yamada, A. and Oiwa, Y.: Priority Integration for Weighted Combinatorial Testing, *Proc. 39th Annual International Computer Software and Applications Conference* (*COMPSAC*), pp.242–247, IEEE (2015).

[2] Colbourn, C.J. and McClary, D.W.: Locating and detecting arrays for interaction faults, *Journal of Combinatorial Optimization*, Vol.15, No.1, pp.17–48 (2008).

[3] Cox, D.R.: The regression analysis of binary sequences, *Journal of the Royal Statistical Society, Series B* (*Methodological*), pp.215–242 (1958).

[4] Do, H., Elbaum, S. and Rothermel, G.: Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact, *Empirical Software Engineering*, Vol.10, No.4, pp.405–435 (2005).

[5] Ghandehari, L.S.G., Lei, Y., Xie, T., Kuhn, R. and Kacker, R.: Identifying failure-inducing combinations in a combinatorial test set, *Proc. 5th IEEE International Conference on Software Testing, Verification and Validation* (*ICST*), pp.370–379 (2012).

[6] Konishi, T., Kojima, H., Nakagawa, H. and Tsuchiya, T.: Finding Minimum Locating Arrays Using a SAT Solver, *Proc. 10th IEEE International Conference on Software Testing, Verification and Validation Workshops* (*ICSTW*), pp.276–277 (2017).

[7] Kuhn, D.R., Kacker, R.N. and Lei, Y.: *Introduction to combinatorial testing*, CRC Press (2013).

[8] Martínez, C., Moura, L., Panario, D. and Stevens, B.: Locating errors using ELAs, covering arrays, and adaptive testing algorithms, *SIAM Journal on Discrete Mathematics*, Vol.23, No.4, pp.1776–1799 (2009).

[9] Nagamoto, T., Kojima, H., Nakagawa, H. and Tsuchiya, T.: Locating a Faulty Interaction in Pair-wise Testing, *Proc. 20th IEEE Pacific Rim International Symposium on Dependable Computing* (*PRDC*), pp.155–156 (2014).

[10] Nie, C. and Leung, H.: A survey of combinatorial testing, *ACM Computing Surveys*, Vol.43, No.2, p.11 (2011).

[11] Nishiura, K., Choi, E. and Mizuno, O.: Fault Localization of Combinatorial Testing with Logistic Regression, Proc. of FOSE（第 23 回ソフトウェア工学の基礎ワークショップ），pp.243–244, JSSST (2016).

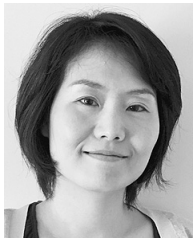
[12] Nishiura, K., Choi, E. and Mizuno, O.: Improving Faulty Interaction Localization Using Logistic Regression, *Proc. 2017 IEEE International Conference on Software Quality, Reliability & Security* (*QRS*), pp.138–149 (2017).

[13] Yilmaz, C., Cohen, M.B. and Porter, A.A.: Covering arrays for efficient fault characterization in complex configuration spaces, *IEEE Trans. Software Engineering*, Vol.32, No.1, pp.20–34 (2006).

[14] Zhang, J., Ma, F. and Zhang., Z.: Faulty interaction identification via constraint solving and optimization, *Proc. 15th International Conference on Theory and Applications of Satisfiability Testing* (*SAT*), pp.186–199 (2012).

西浦 生成

平成 28 年京都工芸繊維大学工芸科学部情報工学課程卒業．学士（工学）．同大学院工芸科学研究科情報工学専攻博士前期課程在学中．平成 28 年国立研究開発法人産業技術総合研究所リサーチアシスタント．ソフトウェアの不具合特定に関する研究に従事．



崔 銀惠 （正会員）

平成 14 年大阪大学大学院情報数理系専攻博士後期課程修了．博士（工学）大阪大学．同年（株）東芝研究開発センター．平成 16 年より産業技術総合研究所特別研究員，研究員を経て，現在，主任研究員．主にソフトウェアのテスト，検証，不具合特定に関する研究に従事．

水野 修 （正会員）

平成 11 年大阪大学大学院基礎工学研究科助手．平成 13 年博士（工学）大阪大学．平成 22 年京都工芸繊維大学大学院工芸科学研究科准教授．平成 29 年同大学情報工学・人間科学系教授．主にソフトウェアのバグ検出手法，ソフトウェアリポジトリのマイニングに関する研究に従事．