機械学習による不具合組み合わせ特定への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)]

概要：不具合組み合わせ特定とは，組み合わ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み合わせを特定する問題である．我々は，機械学習を用いた不具合組み合わせ特定に取り組んでいる．本研究では，まず，組み合わせテストケースに含まれるパラメータ値の組み合わせとテスト結果の成否を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た回帰係数値から，各パラメータ値の組み合わせが不具合組み合わせである疑わしさを決定する．次に，各パラメータ値の組み合わせの疑わしさから，それが不具合組み合わせであるか否かを自動分類するために，境界値決定法，最大距離分割法，*K*-means 法の 3 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法を適用した比較評価実験を行うことで，提案法の有効性を示す．

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．組み合わせテスト [6,10] はブ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み合わ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，組み合わ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み合わせ（Faulty Interaction，以下 FI と呼ぶ）を特定することを「不具合組み合わせ特定」と呼ぶ．

テストの実行結果の成否から不具合組み合わせを特定する手法の一つに，テスト成否から容易に不具合組み合わせを特定可能なテスト（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] で，後者の不具合特定手法として，組み合わ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，ある組み合わせが 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る．学習によって 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求めることで，各説明変数の重み (回帰係数) を算出する．この重みが，各説明変数の持つ目的変数の値決定への寄与度を表すことを利用し，何らかの事象と因果関係にある要因の影響力を数値的に分析することができる．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，組み合わせテストに含まれる大きさ t の組み合わせ全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．これによって各組み合わせの存在がテスト失敗に与える影響の大きさ，すなわち，各組み合わせが FI である可能性を数値として得ることが可能になる．

本論文では，更に，各組み合わ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 3 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み合わせ特定の精度を比較評価する．一つ目は，固定的な基準値を設けてそれ以上の分析値を持つ組み合わせを FI とする方法（M1），二つ目は，正の分析値を持つ組み合わせのうち，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），三つ目は，機械学習の *K*-means 法を用いてクラスタリングを行い FI を決定する方法（M3）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

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
2 産業技術総合研究所 情報技術研究部門
3 京都工芸繊維大学 情報工学・人間科学系
a) k-nishiura@aist.go.jp
b) e.choi@aist.go.jp
c) o-mizuno@kit.ac.jp

表 1 例題システムモデル (System Under Test)

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)	

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み合わ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み合わせテスト生成ツール pricot [1] へ入力して生成した，二項目間の組み合わせを網羅するペアワイズテストスイートの二種類を用いた．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．

提案法では，t 個のパラメータからなる FI（以降，t-FI と呼ぶ）を特定したい場合，まず，組み合わせテストに含まれる大きさ t の組み合わせ（以降，t-tuple と呼ぶ）を全て抽出する．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，クラスタリング手法を用いて 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 と，我々の提案する 3 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通りである．2 章では，本研究の対象である組み合わ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，クラスタリングについて説明する．3 章では，提案する不具合組み合わせ特定法のプロセスについて説明し，4 章では，提案法を適用した評価実験の実施について，5 章では，実験結果と考察について述べる．また 6 章では追加実験について述べる．7 章で関連研究について述べ，最後に，8 章で，まとめと今後の課題を述べる．

2. 準備

2.1 組み合わせテストと不具合特定

組み合わせテスト [6, 10] は，システムのパラメータが取る値の組み合わせに注目して，その組み合わ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．組み合わ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，制約式の三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 をもち，各パラメータは 2 つまたは 3 つの値をもつ．また，$(P_3 = v_6) \rightarrow (P_2 \neq v_3)$ という制約をもち，$P_2 = v_6$ と $P_3 = v_3$ の組み合わせは許されないことを表す．

表 2 パラメータ値のペアの例

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（$\mathcal{T}_1$）の例

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_6
7	v_2	v_3	v_7

表 4 全組み合わせテスト（$\mathcal{T}_2$）の例

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

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．

すべての組み合わせを網羅するテスト（「全組み合わせテスト」，*All combination test*）は，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，ある組み合わせ数 t（*Interaction strength* と呼ぶ）を設定し，t 個のパラメータ間で取り得る値の組み合わせ（以降，t-tuple と呼ぶ）を全て網羅するテスト（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 テストはもっとも広く用いられており，「ペアワイズテスト」と呼ばれる．表3の $\mathcal{T}_1$ は，表 1のシステムモデルに対するペアワイズテストの例であり，制約を満たす全てのパラメータ値のペア（表 2）を網羅している．一方，表 4の $\mathcal{T}_2$ は，例題シ

表 5 ペアワイズテストとテスト結果の例

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

ステムモデルの 3-way テスト，かつ，全組み合わせテストである．

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，組み合わせ数 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「不具合組み合わせ特定」問題である．不具合組み合わせは 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み合わせは複数存在し得ると仮定する．

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，全組み合わ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．

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組合わせテストのテストスイートに含まれる各タプルの疑わしさ（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．

2.3 クラスタリング

クラスタリングはクラスタ分析とも呼ばれ，与えられたデータを外的基準なしに自動的に分類する手法である．機械学習のうち教師なし学習に属する．与えられたデータセットを多次元空間上の点と定義し，データ間の距離やデータ自体の性質を用いて類似性のあるデータをひとかたまりのクラスタにまとめる処理を行う．

クラスタリングを行うアルゴリズムのうち，非階層型のアルゴリズムで最も有名なものが K-means 法 [7] である．K 個のクラスタに分割することが分かっている場合に利用することができる．アルゴリズムの内容を次に示す．

(1) 各データ $x_i(i = 1 \ldots n)$ に対して K 個のクラスタをランダムに割り振る．
(2) 割り振ったデータをもとに各クラスタの中心 $V_j(j = 1 \ldots K)$ を計算する．
(3) 各 x_i と V_j との距離を求め，x_i を最も近い中心のクラスタに割り当て直す．
(4) 上記の処理で全ての x_i のクラスタ割り振りが変化しなくなるまで繰り返す．

3. 提案手法

この章では，組み合わせテストの不具合組み合わせ（FI）を推定する提案法を説明する．提案法の入力は，(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．提案法は大きく次の 2 つのステップに分かれる．

Step 1　疑わしさ計算．テストスイートに含まれる各組み合わせ（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み合わせが FI であるか否かを 3 つのクラスタリング手法のいずれかで決定する．

以下では，例題を用いながら，提案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

表 6　入力例から作成された存在表

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

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，全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，全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 のタプルを 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　Step 1：疑わしさ計算

3.2.1　存在表の作成

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三段階の手順で行う．

(1) テストスイート内に出現する全ての t-tuple を抽出する．
(2) 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) 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のテストスイートと実行結果と $t = 2$ を入力とした場合，全ての 2-tuple を抽出して作成した存在表を表6に示す．この例では 2-tuple は全部で 15 個存在する．

3.2.2　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．

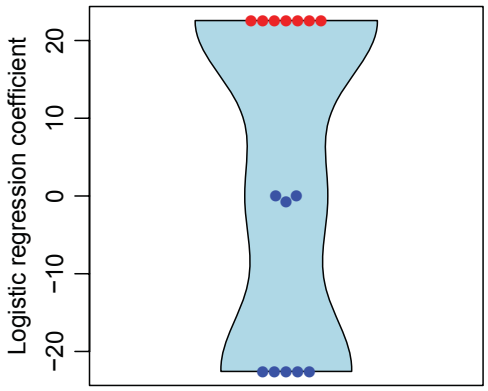

図 1　入力例に含まれる 2-tuple の分析値の分布

表7に入力例に対してロジスティック回帰分析を適用することで得られた 2-tuple の分析値の一覧を示す．また，図1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．

3.3　Step 2：FI 決定

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み合わせが 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 3 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：固定的な基準値を設けてそれ以上の分析値をもつものを 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．

表 7 結果例：各 2-tuple の分析値一覧と提案手法 M1, M2, M3 による FI 分類

2-tuple	回帰係数	M1	M2	M3	real-FI
(v_3,v_7)	22.56	FI	FI	FI	FI
(v_1,v_5)	22.56	FI	FI	FI	FI
(v_3,v_5)	22.56	FI	FI	FI	FI
(v_1,v_6)	22.56	FI	FI	FI	FI
(v_1,v_7)	22.56	FI	FI	FI	FI
(v_1,v_4)	22.56	FI	FI	FI	FI
(v_2,v_3)	22.56	FI	FI	FI	FI
(v_1,v_3)	22.56	FI	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	-	-	-	-

M3 *K*-means によるクラスタリング：正の範囲で *K*-means 法を用いてクラスタリングを行い，FI を決定する．

それぞれの方法について以降の各章で説明する．

3.3.1 M1：閾値決定法

最も簡単な方法として，閾値となる基準値を設けてそれ以上の分析値をもつ *t*-tuple を FI とする方法が考えられる．ここで，基準値は 4.605 とする．その理由を以下に示す．

ロジスティック回帰分析によって得られる分析値，すなわち回帰係数値は，機械学習のプロセスによって得られたロジット比（対数オッズ比）の近似値である．オッズは，ある事象が起こらない確率に対する，その事象が起こる確率の比を表す．オッズは確率の別な表現であるともいえる．オッズと確率 p の関係を式 2に示す．

$$Odds = \frac{p}{1-p} \tag{2}$$

また，オッズ比は 2 つのオッズの比である．対数オッズ比は，オッズ比の自然対数であり，ロジット比とも呼ばれる．

ロジスティック回帰分析において，ある説明変数 x_n の回帰係数 β_n が近似する対数オッズ比を式3に示す．

$$\beta_n \approx \log \frac{\frac{P(y=1|x_n)}{1-P(y=1|x_n)}}{\frac{P(y=1|\overline{x_n})}{1-P(y=1|\overline{x_n})}} \tag{3}$$

ここで，ロジスティック回帰分析における回帰係数は，「他の説明変数をそのままに説明変数 x_n が 1 増加したときの目的変数 y の上昇量」の対数に近似する値となる．これを我々の手法における各組み合わせの存在とテストの成否を対応付けたモデルにおいて説明すると，「説明変数が 1 増加する」という事象は，ある組み合わせの存在を表すブール変数が 0 から 1 になる場合のみを指すので，「ある組み合わせ X がテストケース中に存在したときにテストが失敗する確率は，X がテストケース中に存在しないときにテストが失敗する確率の何倍であるか」と解釈できる．この値を P とすると，P の対数の近似値が回帰係数値としてロジスティック回帰分析値によって得られる．

我々はある組み合わせが FI であるかを決定する基準として，「その組み合わせが存在するテストケースは，その組み合わせが存在しないテストケースよりテストが失敗する確率が 100 倍より大きい」という指標を定める．回帰係数は P の対数に近似することから，次のように回帰係数の満たす数値が求められる．

$$e^{\beta_n} \approx P \geq 100$$

$$\beta_n \geq \log 100 = 4.605170... \tag{4}$$

以上によって，分析値の基準値を 4.605 とする．

たとえば，表7に示す各組み合わせの分析値一覧に対して，4.605 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．

3.3.2 M2：最大距離分割法

提案法 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 とそれ以外の組み合わ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 8 提案手法 M2 によるクラスタリング

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

表7の例題に提案法 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．

3.3.3 M3：*K*-means を用いたクラスタリング

最後に，最も有名なクラスタリング手法の 1 つである *K*-means 法を用いる FI 決定法を提案する．*K*-means 法によってクラスタリングを行う場合，入力として求められるのはクラスタリングしたいデータセットとクラスタ数 K である．つまり，*K*-means 法を使用する場合は事前にクラスタ数を決定しておく必要がある．そこで，クラスタ数 K を 2 とし，提案法 M2 と同じく元データセットから分析値 0 以下のものを除外して分析値 0 のダミーデータを加えたものをデータセットとして用いる．*K*-means によるクラスタリングの結果として各 *t*-tuple に 2 種類のクラスタ番号がラベル付けされたものが得られるので，分析値の高いほうのクラスタに属するタプルを FI とする．

我々の例題に対しては，*K*-means 法によるクラスタリングの結果は表9のようになり，ラベル番号によって分類されたクラスタのうち最も高い水準の回帰係数を持つクラスタに属する 7 つのタプルを抜き出して FI とする．また FI 分類結果は表7の M3 のようになる．

結果として，例として用いた入力に対しては，3 つの提案法 M1, M2, M3 で同じく，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，全ての FI を正しく推定できた結果になっている．

4. 評価実験

我々は Python で提案法の実装を行い，評価実験を行った．実装において，まず，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．また，*K*-means 法によるクラスタリングの実装には，Python の機械学習パッケージ scikit-learn の *K*-means を用いた．

表 9 *K*-means 法 (K=2) によるクラスタリング

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

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み合わ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．全組み合わせテストスイートは全ての入力の組み合わ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 つのパラメータの取り得る値の組み合わせを網羅したテストスイートである．

全組み合わせテストスイートは SIR から入手できる．SIR から入手した全組み合わ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全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．

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 2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10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 に対して g_i 個の値をもつ 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り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 に対して l_j 個のリテラルをもつ h_j 個の節があることを表す．

[*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 2 flex のテストプランの一部

表 10 flex，grep，make のシステムモデルサイズ

Proj.	Model size
flex	パラメータと値のサイズ 29; $3^{23}4^{4}6^{2}$
	制約のサイズ 97; $2^{712}22^{1}24^{2}25^{17}26^{9}$
grep	パラメータと値のサイズ 14; $2^{4}3^{1}4^{3}5^{1}6^{1}9^{1}11^{1}13^{1}20^{1}$
	制約のサイズ 87; $2^{433}3^{27}4^{8}7^{5}16^{1}24^{1}27^{1}28^{1}31^{10}$
make	パラメータと値のサイズ 22; $2^{2}3^{12}4^{4}5^{2}6^{1}7^{1}$
	制約のサイズ 79; $2^{526}21^{1}22^{1}23^{1}24^{3}25^{7}26^{9}$

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，全てのテストケースの実行が成功したバージョンと全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み合わせ不具合特定の対象として不適当であるため除外し，全組み合わ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 11，表 12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 の総数を示す．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．

表 11 全組み合わせテストの諸パラメータ

Proj.	#ver	#tc	#1-tuple	#2-tuple
flex	31	500	48	213
grep	9	440	43	433
make	2	768	32	179
All	42			

表 12 ペアワイズテストの諸パラメータ

Proj.	#ver	#tc	#1-tuple	#2-tuple
flex	31	27	48	213
grep	9	45	43	433
make	2	8	32	179
All	42			

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．そのため実験対象の各バージョンについてあらかじめ real-FI を特定し，比較用の正解データとして用意しておく必要がある．real-FI の特定には従来の原始的アプローチを用いる．すなわち，ある組み合わせについて，それが失敗したテストケースに含まれ，かつ成功したテストケースに含まれていないかどうか探索することで FI であるか否かを判定する．

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，提案法によって FI と分類されたものは全て 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，提案法によって FI と分類されたものは全ての real-FI を含んでいる．正確度は全ての *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，提案法は real-FI と非 real-FI を完全に分離できている．

5. 実験結果と考察

5.1 実験結果

各テストタイプ（すなわち，全組み合わせテストスイート及びペアワイズテストスイート），各 *t*-tuple（すなわち，1-tuple 及び 2-tuple）をそれぞれ対象に行った提案手法の精度評価実験における，使用したすべてのバージョンでの測定評価値の平均値を表13に示す．また，全てのテストタイプ・*t*-tuple での平均値を下段の 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．

表 13 実験結果のまとめ：提案法の精度（statsmodels を用いた場合）

		Precision (%)			Recall (%)			Accracy (%)		
	t	M1	M2	M3	M1	M2	M3	M1	M2	M3
Pairwise	1	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
	2	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
All	1	94.85	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	98.57	**100.00**	**100.00**
	2	96.75	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	99.62	**100.00**	**100.00**
Avg.		97.90	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	99.56	**100.00**	**100.00**
#(100%)		126	**143**	**143**	**143**	**143**	**143**	126	**143**	**143**

ここで求めた平均値は，比較対象である real-FI が存在しないバージョンを除いたものとなっている．これは，対象としているスコープ内に real-FI が存在する場合に我々の提案手法がそのタプルを FI として推定する能力の精度を提示するためである．スコープ内に real-FI が存在しない場合，例えば 2-tuple より大きい real-FI のみを持つバージョンに対して 1-tuple のスコープで FI を推定しようとしたときに，1-tuple の real-FI が存在しないにも関わらず，我々の手法においては少なくとも M2 および M3 を使用した際に必ず FI と推定される *t*-tuple が出現してしまうため，精度が下がってしまう．real-FI がスコープ内に存在しないバージョンは，全組み合わせテストスイートの 1-tuple で 14，全組み合わせテストスイートの 2-tuple で 4，ペアワイズテストスイートの 1-tuple で 7 だけ存在し，ペアワイズテストスイートの 2-tuple では存在しない．すなわち，平均値の集計に用いたバージョン数は $42 \times 4 - (14 + 4 + 7) = 143$ となる．また#(100%) の上限値も 143 となる．

5.2 考察

実験結果から，本手法は極めて高い精度で real-FI を特定できていることがわかる．特に，FI 決定法 M2 および M3 では，適合率，再現率，正確度ともに，全てのバージョンで 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 M1 についても，再現率は全てのバージョンで 100%であり，適合率，正確度に関しても全 143 バージョン中 126 バージョンで 100%，全バージョンでの適合率の平均値は 97.9%，正確度の平均値は 99.56%と，非常に高い精度で real-FI を特定できている．

M1 で適合率が 100%にならなかった 17 のバージョンについては，その全てにおいて，基準値である 4.605 を上回る分析値を持つ非 real-FI な組み合わせが出現したことが原因となっている．またあくまでも回帰係数は近似値として計算されるので，理論的に求められた基準値と実際に求められた分析値の間にギャップがあるのは当然ともいえる．基準値の決定に際して，今回は 100 倍と定義した部分をたとえば 1000 倍，10000 倍とさらに厳しくしたり，近似のブレを考慮して余裕をもたせた基準値に設定することで，精度が改善される可能性がある．

M2 と M3 はともに 100%の精度となったが，M3 は M2 に比べると計算速度が遅いので，M2 のほうがより優れた方法であると評価することができる．ただし，*K*-means 法の特性を考えると，M2 で正しく分割できる場合については M3 でも正しく分割できるのは当然であるともいえる．今回の実験では全てのバージョンで M2 によって正しく分割可能な分析値分布であったためこのような結果になったが，仮に M2 では正しい分割が難しい分析値分布があったとし，M3 を使用することで正しい分割が可能となる場合には M3 の隠れた優位性が示される余地が残ることに注意したい．

また先に述べたように，表13に示した数値は対象としているスコープ内に real-FI が無い場合のものを除いて集計したものである．これは提案法の real-FI を特定する能力を示すためであるが，実際に提案法を未知の対象物に適用する場合にはスコープ内に real-FI が存在するのかどうかは不明であり，このため FI 決定法 M2 および M3 を用いた場合に誤った FI を選択してしまう問題が残る．逆説的に，スコープ内に real-FI が存在しない場合には「FI は存在しない」と正確な判定が可能な方法 M1 こそが，不具合組み合わせ特定問題において最も適切であると結論付けることもできる．

6. 追加実験

我々の行った評価実験ではロジスティック回帰分析の実行に Python の統計モデルパッケージ statsmodels を使用したが，他のツールを用いてロジスティック回帰分析を実行する場合でも同様な結果になるのか確かめてみることには大いに意義がある．ここでは追加実験として，同じ Python の機械学習パッケージである scikit-learn を使用して同じ実験を行う．追加実験において，ロジスティック回帰分析の実行に使用していた statsmodels を scikit-learn に変更したこと以外，実験対象，実験手順，および結果データの取り扱いは，全て同じである．また scikit-learn のロジスティック回帰を実行する関数 LogisticReggression において全てのパラメータを初期値のまま使用した．これは statsmodels の場合も同じである．結果を表 14に示す．

表 14 追加実験：scikit-learn を用いた場合の提案法の精度

		Precision (%)			Recall (%)			Accracy (%)		
	t	M1	M2	M3	M1	M2	M3	M1	M2	M3
Pairwise	1	0.07	75.89	**76.04**	3.57	63.53	**68.29**	87.02	**89.39**	89.30
	2	0.00	62.33	**63.61**	0.00	14.24	**45.67**	83.40	84.28	**85.94**
All	1	0.00	69.92	**72.84**	0.00	15.49	**39.75**	77.00	79.40	**81.45**
	2	0.00	**74.42**	73.25	0.00	18.44	**39.13**	77.20	80.12	**81.28**
Avg.		0.02	71.90	**72.39**	0.89	42.88	**59.43**	83.33	85.87	**86.83**
#(100%)		1	**80**	45	1	40	**43**	26	**31**	27

表13に示した statsmodels を用いた場合の結果と比べて極端に精度が下がっていることが確認できる．特に FI 分類方法 M1 の場合には適合率と再現率の平均値はともに 1%に満たない．これは M1 の使用している基準値を超える分析値が現れなかったケースが大部分であることに由来する．

また，図3および 4は statsmodels および scikit-learn をそれぞれ用いた場合の，flex の 1 バージョンにおける分析値の分布を表す．statsmodels を使用した場合では 1-tuple, 2-tuple のスコープでともに real-FI と非 real-FI がくっきりと分かれており，M1, M2, M3 の各 FI 分類方法によって FI が正しく決定できそうに見える．一方で scikit-learn を使用した場合では real-FI と非 real-FI の分析値は明確に分かれておらず，極めて連続的であり，M1, M2, M3 のどの FI 分類方法であっても real-FI のみを正しく決定することは不可能に思える．また，statsmodels の場合では 1-tuple, 2-tuple で最大値が 20 付近で固定されているのに対し，scikit-learn の場合には 1-tuple で最大値は 3 付近，2-tuple では最大値は 1 付近と，スケールの変化が起きていることも確認できる．

この追加実験の結果から，我々が前章までで行ったような statsmodels を用いたロジスティック回帰分析による分析値を利用した不具合組み合わせ特定手法は，他のロジスティック回帰分析を実行する方法を用いたときには同様な結果をもたらすわけではないことを示す．この理由として次のようなことが考えられる．

一つは，ロジスティック回帰分析を行うときの諸パラメータの違いによるものである．statsmodels および scikit-learn でのロジスティック回帰分析の実行は全てのパラメータを初期値で使用したが，初期値とされている値が二つの方法間で異なっているという可能性がある．もう一つは，statsmodels および scikit-learn ではそもそもロジスティック回帰を実行するアルゴリズムが異なっていることによるものである．ロジスティック回帰を実現するアルゴリズムには最急勾配法や Newton 法，あるいは統計モデルを用いる方法など複数の種類が存在する．statsmodels と scikit-learn でロジスティック回帰に使われているアルゴリズムが異なる場合，性質の異なる結果が導出されてしまう可能性がある．

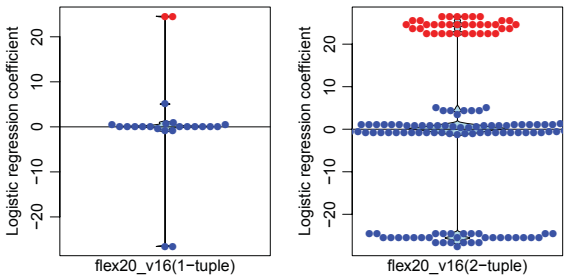

図 3 statsmodels を用いたロジスティック回帰分析における回帰係数分布例

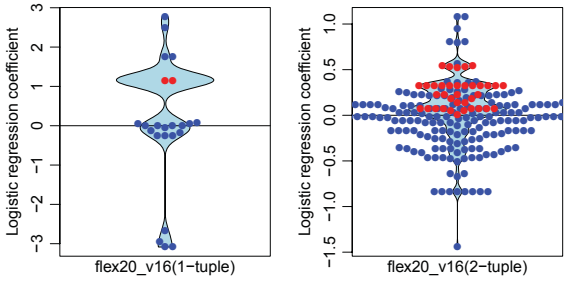

図 4 scikit-learn を用いたロジスティック回帰分析における回帰係数分布例

7. 関連研究

組み合わせテストの不具合組み合わ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二つに分けられる．

Non-Adaptive アプローチには，テスト成否から容易に不具合組み合わせを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 t-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．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，与えられた組み合わせテストの 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，与えられた組み合わ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，更に，FI であるか否かを自動分類するための 3 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，提案法の精度の良さを推測できる．また我々は，FI 決定のスコープ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では提案しており，我々の研究では今後の課題である．

8. 結論

本稿では，ロジスティック回帰分析を用いて組み合わせテストにおける不具合組み合わせの疑わしさを定量的に計算し，機械学習を含む 3 つの方法を用いて分析値の分布から自動的に FI を特定する手法を提案した．また，実際にバグを含むプロジェクトの組み合わせテストとその実行結果を対象に提案法の 3 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，より多様な実験対象で評価実験を行うこと， 他の FI 特定法とのパフォーマンス比較を行うこと，などが挙げられる．

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. of the 39th Annual International Computer Software and Applications Conference (COMPSAC)*, IEEE, pp. 242–247 (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. of 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST)*, pp. 370–379 (2012).

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

[7] MacQueen, J.: Some methods for classification and analysis of multivariate observations, *In Proc. of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1, No. 14, pp. 281–297 (1967).

[8] Martnez, 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. of IEEE 20th Pacific Rim International Symposium on Dependable Computing*, 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 (in Japanese)*, JSSST, pp. 243–244 (2016).

[12] Nishiura, K., Choi, E. and Mizuno, O.: Improving Faulty Interaction Localization Using Logistic Regression, *Proc. of the 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 Transactions on 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. of International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pp. 186–199 (2012).