B-025

組み合わせテスト集合生成におけるテストケース候補数の最適化について
Optimization of the number of test case candidates in generating a combinatorial test suite

†大橋　輝‡　土屋　達弘†
Teru Ohashi　Tatsuhiro Tsuchiya

1. はじめに

ソフトウェアシステム開発では，十分なテストの実施と，テスト工程に掛かる費用の削減という矛盾した要求が生じる．よって少ないテストで多くの状況をカバーする必要がある．これを実現する方法として組み合わせインタラクションテスト(CIT)と呼ばれるテストセット生成法がある．これは少数の機能の組み合わせをテストすることで不具合が検出できると考え，少ないテストで t 組の組み合わせを網羅するテストを行う方法である．このようなテストセットの生成は通常ツールを用いて行う．テスト設計者は，どのような機能，要素に注目するかを判断し，それらの調整とテストセット生成を何度も繰り返しながら，最終的なテストセットを設計する．したがって，ツールによるテストセットの生成は，高速に実行できることが強く求められる．本研究ではこのテスト生成法の最適化を行い，高速かつテストケース数の少ないテストセットの生成を目指す．

2. 既存手法

以下に 4 パラメータ（機能）を有するシステムに対する，t=2 の場合の CIT 用のテスト集合の例を示す．ここでは，OS は３値，その他のパラメータは２値を取るものとする．すべての t=２個のパラメータについて，値の組み合わせが 6 個のテストケースのいずれかによってカバーされている．

#1	Windows 7	IE	IPv4	MySQL
#2	Windows 7	FF	IPv6	Sybase
#3	Windows 8	IE	IPv6	Sybase
#4	Windows 8	FF	IPv4	MySQL
#5	Vista	IE	IPv4	Sybase
#6	Vista	FF	IPv4	MySQL

CIT 用のテスト生成方法として，貪欲手法が広く用いられている．これは，テストケースを一つずつ生成していき，最終的にすべての組み合わせがテスト集合に現れた段階で生成を完了する手法である．

各テストケースはヒューリスティックを用いて生成する．これは未出現の組み合わせを最大数含むテストケースを生成する問題は NP 困難なためである．多くの貪欲手法に基づくアルゴリズムでは，幾つかの候補を生成し，その中から現在未出現の組み合わせが最も多く現れる候補を選択する．このとき，候補を増やすほど良いテストケースが選択でき，結果として小さいテスト集合を作成することができる．一方，候補を増やすとその分生成にかかる時間が増大してしまう．

Teru Ohashi, Tatsuhiro Tsuchiya
†大阪大学　情報科学研究科　Graduate school of Information Science and Technology, Osaka University

3. 提案手法

従来，各テストケースを選択するにあたり，生成するテストケース候補の数は固定値(以降 M と表記)として扱われてきた．本研究では，テストセットの生成過程でこの数を変化させることで，テストセットの大きさと生成時間の最適化を行う．具体的には，生成の前半と後半とで異なる候補数を用いる．

この方法の着想は次の予想に基づく．まず，初期段階では未出現の組み合わせが多くあるため比較的容易に多くの組み合わせを含むテストケースを見つけることができる．一方，生成の末期になると，未出現の組み合わせが少なくなるため，それらを同時に多く含むテストケースの発見がより困難になると考えられる．よって，生成の初期では候補数を小さくして計算コストを消費しないことで，テストセットの大きさを小さく保ったまま，生成時間を削減できることが期待できる．

提案法を具体化するにあたって，まず M1, M2, θ という値を定義する．M1 は前半に用いる候補数である．M2 は計算後半に用いる候補数である．最後に前半と後半を区別するため，テストセットがどの程度まで構築されているかを表す指標 θ を以下の様に定義する．カバーする組み合わせの総数を UCI，その内，現在のテストセットによってカバーされていない数を UC とし，θ = UC/UCI と定義する．θ が 0.1 以下の時は全テストケース中残っている未カバーテスト数は 1 割であると考えられる．本研究では，θ が一定以下となった時にテストケース候補生成数を M1 から M2 に切り替えることで，上記の提案を実現する．

4. 実験,評価

実験は既存のツールである CIT-BACH を改変しておこなった．まず，M1, M2 の値については，CIT-BACH では M=20 としていること，文献[1]では，M として 1, 5, 10 の値を用いていることから，M1 は 1，5，10 の 3 通りとし，M2 は 20 として，実験を行った．また，M = 1, 5, 10, 20 の 4 通りについて，候補数を変化させない場合についても実験を行った．実験では，t=2 とし，ベンチマークとしてしられている apache, gcc, spinv, bugzill, spins の 5 つの問題例を使用した．各設定に対し 20 回テスト生成を行い，テストセットの大きさと実行時間についての平均値を求めた．実験結果を以下の図に示す．ここでは apache, gcc, spinv の結果を載せる．

図では，M を 1 , 5, 10, 20 に固定した場合と，θ ＝0.1 として M2 と 20，M1 を 1, 5, 10 とした場合，および，θ ＝ 0.05 として同様に実験した場合の結果を示している．

まず，M を固定した場合，テスト候補数を増やせばテスト数を少なくすることができる一方，テスト生成時間は増加することが分かった．ただし，生成時間は M にほぼ比例して増加するが，テスト数への影響は M が 10 を超えると目立たなくなった．

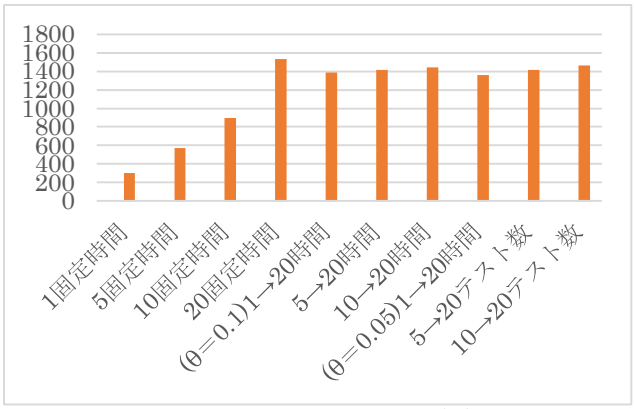

図 1 ：Apache 平均テスト生成時間

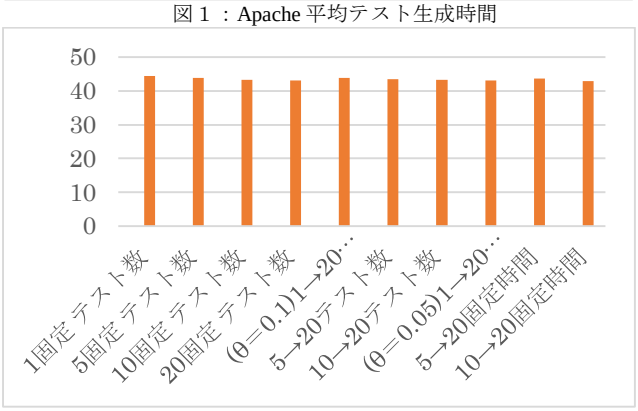

図 2：Apache 平均テスト生成数

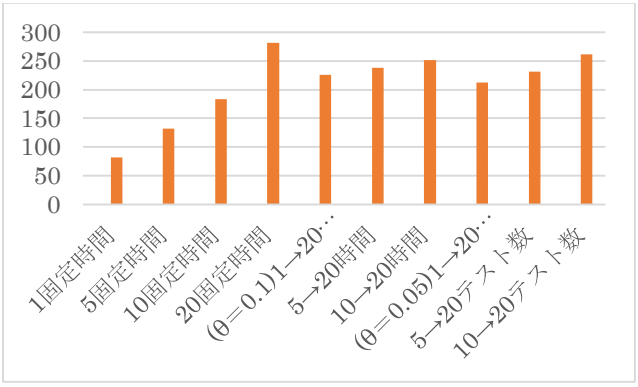

図 3：spinv 平均テスト生成時間

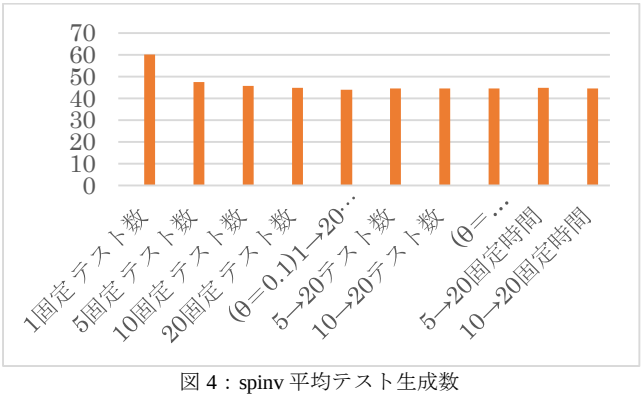

図 4：spinv 平均テスト生成数

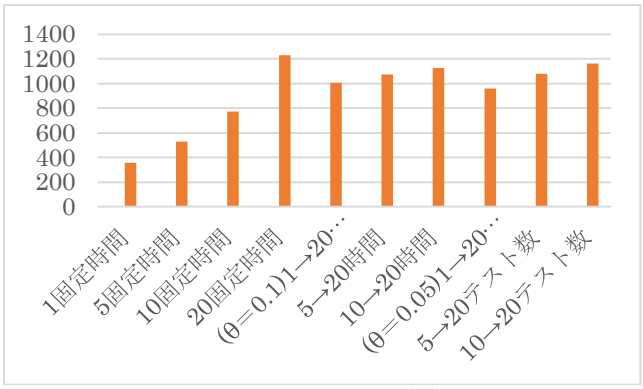

図 5：gcc 平均テスト生成時間

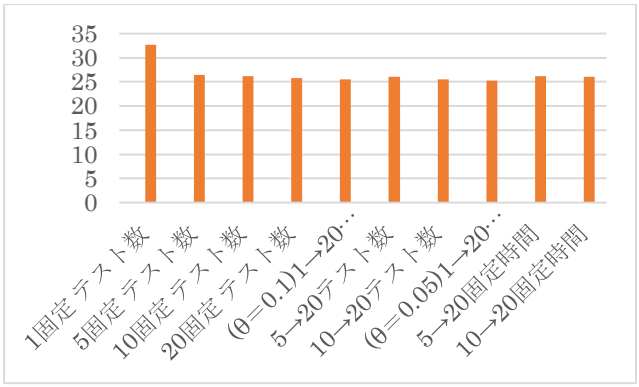

図 6：gcc 平均テスト生成数

一方 M を変化させた場合，Apache, spinv においては，M を 20 に固定するより少ない生成時間で，かつ，差は僅かではあるがテスト数の少ないテストセットを生成できた．gcc についても θ =0.1 の場合は，同様の結果を得ることができ，θ =0.05 でも固定時より少ない時間でほぼ同サイズのテスト集合が生成できた．また，ほぼ全ての場合において，θ は 0.05 より 0.1 とした方が短い時間でテスト生成が終了した．これは最終的に生成するテストケースの数が減少したためと考えられる．

5. おわりに

従来のテストケースの候補数 M を固定した時と比べて，途中で M を変化させることにより生成時間を削減することができた．また，それに加え，多くのベンチマークで 20 へ固定した場合より小さいテストセットを生成することができた．今後の課題としては，より効果の高い変化方法の開発が挙げられる．

謝辞

本研究の一部は JSPS 科研費 15K00098 の助成を受けて実施した．

参考文献

[1] Bryce, R.C.; Colbourn, C.J.; Cohen, M.B., “A framework of greedy methods for constructing interaction test suites,” Proceedings. 27th International Conference on Software Engineering (ICSE 2005), pp.146-155 (2005).