# eテスティングにおける最大クリーク抽出法を用いた等質テスト生成数の最大化法

A Method to extract the maximum number test forms Using MaxClique

石井 隆稔[*1] ソンムァン・ポクポン[*1] 植野 真臣[*1]
Takatoshi ISHII Pokpong SONGMUANG Maomi UENO

[*1]電気通信大学大学院　情報システム学研究科
Graduate school of Infoemation Systems, The University of Electro-Communication.

This paper proposes tan algorithm which generates parallel tests from one item bank. The feature of this method is to be able to maximize the number of parallel tests which permits items overlapping among the tests using maxclique. Some simulation experiments show the effectiveness of this proposal.

## 1. はじめに

実際のテスト構成では, しばしば複数の等質なテストの生成が必要となる. 例えば, 以下のような状況があげられる.

1. 資格試験などにおいて, 毎回のテストで合格の難易度が異ならないようにテスト得点分布および所要時間分布などテストの属性がほぼ一定でなければならない.

2. 入試などで選択科目や選択問題がある場合、それぞれのテスト得点分布および所要時間分布などテストの属性がほぼ一定でなければならない.

3. 追試などで別の試験を出題するときテスト得点分布および所要時間分布などテストの属性がほぼ一定でなければならない. 例えば, 遅刻してきた生徒に対して, そのほかの生徒と同一の能力を測るために別の試験を行ったり, 隣り合う席で別々の試験を行いたいときなどである.

これまで, 等質なテストは作問者の勘と経験によって構成されてきた. しかし, 近年,e テスティングの普及に伴い, アイテムバンク方式のテスト生成が一般化し, 等質テストの複数生成の自動化が可能になりつつある.

Ackerman[1] はヒューリスティックな方法を基にした等質テストの複数生成法を提案している. しかし, この手法は生成されたテスト属性と目標との誤差が大きいという問題があった.

そのため,Luecht と Hirsch[2] は, テストを構成している最中のテスト属性と目標との差を考慮しながらテストに選ぶ項目を選択する改善を提案している。後に Luecht[3] はこの方法を拡張してさらに誤差を小さくしている.

Armstrong,Jones & Kunce[4] は, 等質テストの複数生成をネットワークフロー問題を用いて解くことを提案している. この手法の特徴は　考えられる組み合わせの中で誤差が最小になる組み合わせを選び出せる点である. しかし, 一度にすべてのテストの生成を行うため, 最適化のパラメータの数が非常に多くなり, 計算時間が莫大となってしまう問題がある.

連絡先: 電気通信大学大学院　情報システム工学科, 〒 182-8585
東京都調布市調布ヶ丘 1-5-1

Linden[5],[6] は, 同じく等質テストの複数生成を線形計画法を用いて行っている. 彼は Armstromng の研究 [4] で問題となった変数の数を抑えて等質なテストを複数生成することに成功している. この手法は段階的にテストを選び出す手法で, 生成されるテストと, 残りの項目集合の属性を等質にするというものである.

これまで紹介した手法では生成されるテスト数を最大化している保障がないという問題があった. それを解決するために Belov[7] は生成テスト数の最大化を行うアルゴリズムを提案している. このアプローチは集合充填問題 (Set Packing) を用いて複数テスト生成を行うものである. しかし, アイテムバンクを排他的な部分集合に分割し、それをテストとする手法であるためテスト間の項目の重複が許されないという制約があった. そのため, 一つのアイテムバンクから生成されるテスト数に制限があり, アイテムバンクを十分有効活用できないという問題があった.

本研究では最大クリーク抽出法を用いて生成テスト間に項目重複を許す場合についても生成テスト数を最大化するアルゴリズムを提案する. 従来手法のアプローチとの大きな差異は

1. 生成テスト間に項目重複を許す場合でもテストを生成できる点,

2. 複数テスト生成を最大クリーク抽出法を用いて解く点,

である.

その結果, 提案手法の利点としては

1. テスト間の項目重複を許しているため, 許さない場合よりもより多くのテストを生成でき, アイテムバンクをより有効活用できる

2. 一つのアイテムバンクから生成するテスト数が最大になることが保証できる

が挙げられる. 更に, シミュレーション実験を行い, 本提案の有効性を示す.

## 2. テスト管理システム (TMS:Test Management System)

著者の研究室では、これまで e テスティングを管理するための TMS(Test Management System) を開発してきた (図 1, 文献 [8]). 本システムは (1) 項目生成支援システム (Item Authoring System), (2) アイテムバンク (Item Bank), (3) テスト実施システム (Test Delivery System), (4) テスト構成支援システム (e-Testing Construction Support System), (5) テストデータベース (Test Database), (6) テスト分析システム (Data Analysis System) からなる. 本手法は (4) テスト構成支援システム (e-Testing Construction Support System) として実装する。

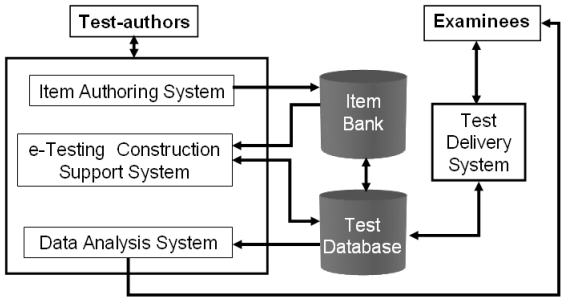


図 1: TMS(Test Management System) の構成

(4) テスト構成支援システム (e-Testing Constraction Support System) は, アイテムバンクから自動的に項目を選択しテスト生成を行う機能と, テスト構成者が, テスト得点・所要時間分布などを予測しながら対話的にテスト構成支援するシステムを持つ.

自動テスト生成機能では、目標となる得点分布や所要時間分布などの条件を入力すると、その条件に合うテストを自動的に生成することができる。これにより、テスト生成者は一度に大量のテストを生成することが可能になる。本論文のテスト構成手法はこのモジュールで使用される。

## 3. 最大クリーク抽出を用いたテスト構成法

### 3.1 領域テスト構成への分割

自動テスト構成とは, テストの出題領域や平均正答率などの様々な条件を入力することで, システムが自動的にその条件を満たすテストを構成する仕組みのことをいう.

条件にはテスト情報量の最大化などに代表される最適化条件も含まれ, 例えば,1800 の要素をもつアイテムバンクから 100 項目のテスト 1 セットを抽出する場合、考えられる項目の組み合わせ (テスト) は ${}_{1800}C_{100}$ となる.

この問題を解決するために, これまでの先行研究 (例えば Belov[7]) では、アイテムバンクが項目の分野や領域によって分類されている細目標構造を持っていることを利用し, 各領域ごとに分割して領域テストを構成し, 最後に統合する方法が一般的に使用されてきた. 本論でもこれに従い, アイテムバンクにおける各領域ごとにテスト条件を満たすテストを構成する仕組みを考える.

### 3.2 アルゴリズム

アルゴリズム全体の流れは以下のとおりである

**STEP.1** 領域テスト構成への分割
与えられたアイテムバンクとテスト構成条件を、領域テストごとのアイテムバンクと構成条件に分割する.

**STEP.2** 領域テスト構成
それぞれの領域テストについてテスト構成を行っていく. 領域テストの構成アルゴリズムについては後述

**STEP.3** 領域テスト統合
生成された領域テストを統合し最終的なテスト群を生成する.

領域テスト構成アルゴリズムについて述べる. ただし, 本アルゴリズムであらかじめ入力される領域テストの生成条件は以下のとおりである.

1. 等質情報量分布の上限下限
2. 平均得点
3. 平均所要時間
4. 生成される等質テストが持つ重複項目数の上限

図 3 に具体的な擬似コードを示す. ただし, 領域テストは項目の集合である. SUBSET($V$) は集合 $V$ の領域集合全体とする. Constraint($t$) はテスト $t$ が与えられたテスト構成条件を満たしているかいないかを返す関数で, 式 (1) で定義されるものとする.

$$\text{Constraint}(t) = \begin{cases} 1 & \text{テスト } t \text{ が条件を満たす} \\ 0 & \text{それ以外} \end{cases} \quad (1)$$

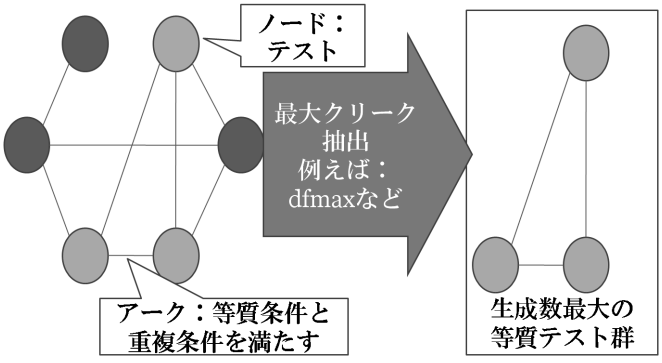


図 2: 領域テスト構成と最大クリーク

STEP.1 ではテスト間の項目重複については考慮せず, アイテムバンクから構成できるすべての組み合わせの中から属性条件を満たすテストをすべて列挙し, それをグラフの頂点とみなしている.

領域テスト構成アルゴリズム入出力

**INPUT:** 入力
$Bank$ \\ 分野ごとに分割された項目の全体集合
Constraint($t$) \\ テスト構成条件関数
$overlap$ \\ 重複限界数
MAXCLIQUE($G$) \\ グラフ G 最大クリーク

**OUTPUT:** 出力
$TestSet$ \\ 生成された領域テストの集合

領域テスト構成アルゴリズム擬似コード

**STEP.1** 頂点の発生
\\ 頂点集合の初期化
$V = \phi$
**for all** $t \in$ SUBSET($Bank$) **do**
**if** Constraint(t) = ture **then**
$V := V \cup \{t\}$
**end if**
**end for**

**STEP.2** エッジの生成
\\ エッジ集合の初期化
$E = \phi$
**for all** $u, v \in V$ **do**
**if** $u \neq v$ **then**
**if** $|u \cap v| \leq overlap$ **then**
$E := E \cup \{(u, v)\}$
**end if**
**end if**
**end for**

**STEP.3** 最大クリークの抽出
$G := (V, E)$
$TestSet :=$ MAXCLIQUE($G$)

図 3: 領域テスト構成アルゴリズム擬似コード

STEP.2 では,STEP.1 で生成した頂点 (テスト) 間の重複項目数が重複項目数の上限値以下なら, その頂点間にエッジを生成している.

STEP.3 では,STEP.1,2 で生成したグラフから最大クリーク抽出を行っている. これまでで生成したグラフ構造のクリークは, クリーク中のどの二つのテストも重複項目数は上限以下であるという性質を持つ. よって STEP.3 で抽出した最大クリークはその中でも最大の要素数を持つ集合である.

### 3.3 限定法による高速化

生成テスト数の下限は領域テストの生成数の最小値となる.

テストはそれぞれの分野の領域テストをひとつづつ選んで構成される. このとき、領域テストは一回以上選ばれることはないため、領域テスト生成数の最小が全体のテストの生成数となる.

この下限値を用いて高速化を行っている. すなわち、領域テストを構成していく上で領域テスト生成数の最小値を保持し、それ以上の領域テストを発見した時点でその分野の探索は打ち切っている.

また、早い段階で領域テスト生成数の最小値を見つけたほうが計算時間を短縮することができる. そこで、本アルゴリズムでは領域テスト構成の順を解空間のサイズの小さなものから行うこととしている. すなわち、$n$ を領域アイテムバンクサイズ、$m$ をその領域テストの出題項目数として、${}_nC_m$ の小さなものから探索を行っている. これはこの要素数 $n$ のアイテムバンクから出題項目数 $m$ で作り出せるテストのバリエーション全体の数である.

## 4. 実験

### 4.1 従来手法とのテスト生成数の比較

提案手法と従来手法のテスト生成数を比較する. 実際には, 以下のような実験をおこなった.

1. 同一のアイテムバンク, 同一のテスト生成条件を用いて従来手法と提案手法でテストを生成する.
2. 従来手法と提案手法のテストの生成数を比較する.

それぞれの条件で 100 種類のアイテムバンクからテストを生成し, その生成数を比較した. ただし, 従来手法では Linden[5] の手法を用い, その中で使用される線形計画問題のソルバーとしては GLPK[9] を使用した.

表 1: 提案手法と従来手法のテスト生成数の比較

| アイテムバンクサイズ | 項目数 | 情報量下限 | 生成数 | | |
|---|---|---|---|---|---|
| | | | B>MC | B=MC | B<MC |
| 180 | 18 | 0.3 | 0 | 67 | 33 |
| | | 0.35 | 0 | 77 | 23 |
| | | 0.4 | 0 | 78 | 22 |
| | 27 | 0.3 | 0 | 75 | 25 |
| | | 0.35 | 0 | 78 | 22 |
| | | 0.4 | 0 | 90 | 10 |
| 270 | 18 | 0.3 | 0 | 37 | 63 |
| | | 0.35 | 0 | 61 | 39 |
| | | 0.4 | 0 | 77 | 23 |
| | 27 | 0.3 | 0 | 59 | 41 |
| | | 0.35 | 0 | 80 | 20 |
| | | 0.4 | 0 | 85 | 15 |
| 360 | 18 | 0.3 | 0 | 30 | 70 |
| | | 0.35 | 0 | 47 | 53 |
| | | 0.4 | 0 | 63 | 37 |

表 1 は従来手法と提案手法のテスト生成数を表している. “アイテムバンクサイズ” は試行でのアイテムバンクに含まれる項目数を示している. “項目数” はテスト生成条件で定義された生成されるテストの項目数を示している. “情報量下限” はテストの生成条件で定義された生成されるテストの項目平均情報量の下限値を示している. それぞれの条件で 100 回試行を行い, 二つの手法のテスト生成数を比較した. “B>MC” は従来手法が提案手法の生成数よりも多い回数を, “B=MC” は従来手法の生成数が提案手法の生成数と同じである回数を, “B<MC” は従来手法が提案手法の生成数よりも少ない回数を, それぞれ示している.

すべての条件で提案手法のテスト生成数は, 従来手法生成数と同じか, それ以上のテストを生成することができた.

### 4.2 重複の有無による生成数比較

次に, テスト間の重複を許した場合にどの程度生成されるテスト数を向上できるかについて考える. 実際には, 以下のような実験をおこなった.

1. 同一のテスト生成条件を用いて, 重複項目数の上限値を変化させテストを生成する.
2. それぞれの重複項目数の上限値でテストの生成数を比較する.

それぞれの条件でアイテムバンクを変えながら 25 回ずつ試行を行い, テスト生成数を比較した.

表 2 は部分テストの重複項目数とテスト生成数を表している.

表 2: 重複項目数とテストの生成数

| アイテム バンクサイズ | 項目数 | 領域 項目数 | 領域項目 重複数上限 | 生成数 平均 | 生成数 標準偏差 |
|---|---|---|---|---|---|
| 180 | 27 | 3 | 0 | 0.7 | 0.6 |
| | | | 1 | 1.5 | 1.5 |
| | | | 2 | 5.1 | 4.9 |
| 270 | 27 | 3 | 0 | 2.4 | 0.8 |
| | | | 1 | 8.5 | 2.9 |
| | | | 2 | 31.0 | 21.9 |
| 360 | 27 | 3 | 0 | 3.2 | 1.0 |
| | | | 1 | 15.5 | 7.4 |
| | | | 2 | 112.7 | 75.7 |
| 450 | 27 | 3 | 0 | 5.4 | 1.0 |
| | | | 1 | 32.6 | 10.7 |
| | | | 2 | 325.3 | 134.5 |

“アイテムバンクサイズ” は試行でのアイテムバンクに含まれる項目数を示している. “項目数” はテスト生成条件で定義された生成されるテストの項目数を示している. “領域別項目数” はテスト生成条件で定義された生成される領域テストの項目数を示している. “領域別重複数上限値” は生成される 2 つのテストの領域テスト間の重複項目数の上限値を示している. それぞれの条件で 25 回試行を行い, テスト生成数を比較した.

すべての条件でテスト生成数は, 重複を許した場合のほうが許さない場合より大きくなる結果であった.

以上に示したように, 提案手法により多くの等質テストを生成でき, アイテムバンクをより有効に活用できることを示した.

## 5. e テスティング自動構成システム

最後に提案アルゴリズムを筆者の研究室がこれまで開発してきた TMS に実装した.

図 4 がシステムのインターフェイスである. このウィンドウから、テストの生成条件を入力しテスト生成を行う. 生成したテストはさまざまな属性を閲覧することが可能である.

## 6. むすび

本論文では,e テスティングにおけるテスト構成において、テストの生成数を最大化する手法を開発・実装した.

具体的には最大クリーク抽出を使用し、従来手法より多くのテストを生成できることをシミュレーション実験を行い示した. 従来手法は Linden[5] と比較し、同条件下では常に提案手法のテストの生成数が多くなる結果であった。

また、テストの構成条件において、項目の重複を許すことによりテストの生成数を飛躍的に増大させることができることをシミュレーション実験を行い示した. 項目の重複を次々増やしながらテストの生成数を比較した. その結果、項目数を増やすとテストの生成数が爆発的に増加する結果であった.

更に、今後の課題として全体のアルゴリズムの高速化を行い、さらに大規模なテスト構成に対応できるよう改良を加えていきたい.

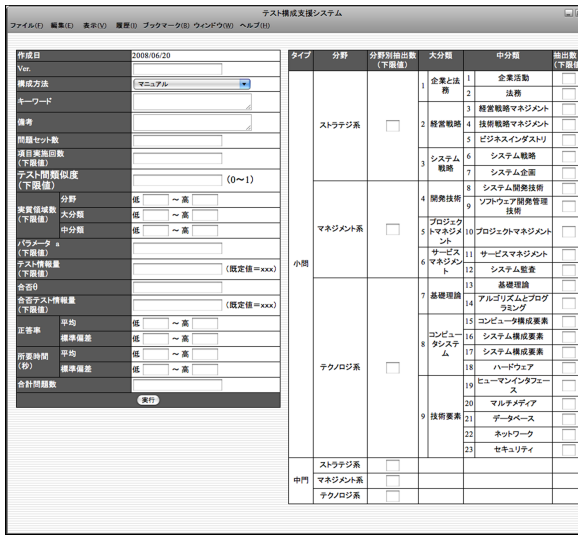


図 4: インターフェイス構成

## 参考文献

[1] Ackerman.T. An alternative methodology for creating parallel test forms using the irt information function. *Paper presented at the annual meeting of National Council on Measurement in Education ,SanFrancisco*, 1989.

[2] Luecht . R.M. & Hirsch . T.M. Computerized test construction using an average growth appoximation of target infomation function. *Applied Psychological Measurement*, Vol. 16, pp. 41–52, 1992.

[3] Luecht.R.D. Computer-assisted test assembly using optimization heuristics. *Applied Psychological Measurement*, Vol. 22, pp. 224–236, 1998.

[4] Armstrong.R.D. Jones.D.H. & Kunce.C.S. Irt test assembly using network-flow programing. *Applied Psychological Measurement*, Vol. 22, pp. 237–247, 1998.

[5] Wim J.van der Linden. *Liner Models for Optimal Test Design*. Springer, 2005.

[6] Jos J. Adema Wim J. van der Linden. Simultaneous assembly of multiple test forms. *Journal of Educational Measurement*, Vol. 35, pp. 185–198, 1998.

[7] Ronald D. Armstrong Dmitry I. Belov. A constraint programming approach to extract the maximum number of non-overlapping test forms. *Computational Optimization and Applications*, Vol. 33, pp. 319–332, 2006.

[8] 植野真臣ソンムァン ポクポン. 統合型 e テスティング・システムの開発と実践. テスト学会誌, Vol. 4, No.1, pp. 53–64, 2008.

[9] *GLPK*. http://www.gnu.org/software/glpk/glpk.html.