修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報理工学研究科　　情報・ネットワーク工学専攻　博士前期課程		
氏　　名	渕本 壱真	学籍番号	2031124
論 文 題 目	Zero-suppressed binary decision diagrams を用いた等質テスト構成		

要　旨

等質テストとは各テストで出題される項目が異なるが，受験者得点の予測誤差が等質なテスト群である．等質テストでは同一能力の受験者が異なるテストを受験しても同一の得点となる保証があり，アイテムバンクから可能な限り多く生成することが望ましい．渕本ら（2020）は最大クリーク問題と整数計画法を用いた並列探索手法を提案した．しかし，整数計画法の時間計算量が大きく，テスト構成数の改善には限界があった．この問題を解決するために，本論文では，場合分け二分木の圧縮表現である ZDD（Zero-suppressed Binary Decision Diagrams; ゼロサプレス型二分決定グラフ）を用いた新たな手法を提案する．ZDD とは場合分け二分木の圧縮表現であり，非循環有向グラフにより組み合わせ集合を表すデータ構造である．コンパクトなデータ構造で組合せ集合を表現できるため，メモリ消費や計算時間を抑えられる可能性がある．本研究では，場合分け二分木の深さ i の節点を項目 i の要素とし，各節点は項目 i を選択する 1-枝と選択しない 0-枝で二分する．この場合分け二分木は全ての項目の組み合わせ（テスト）を評価できる．しかし，時間・空間計算量が大きいため，ZDD を用いてこの場合分け二分木を圧縮する．具体的には，テストの項目数と受験者得点の予測誤差が等質（項目反応理論におけるテスト情報量）な節点を共有し，場合分け二分木を圧縮する．ただし，重複して項目を出題することは項目流出のリスクを増大させ，その項目の特性劣化の原因になることが知られている．そのため，構築した ZDD からランダムサンプリングを行い，任意の二つのテストにおける共通項目数が一定値以下となるように制御する．その結果，従来手法が 1 ヶ月を要しても生成できなかった 50 万以上の等質テストを 24 時間以内で生成できた．

令和三年度 修士論文

Zero-suppressed binary decision diagrams を用いた等質テスト構成

2022 年 1 月 27 日

電気通信大学大学院 情報理工学研究科

情報・ネットワーク工学専攻 情報数理工学プログラム

学籍番号 2031124

渕本 壱真

主任指導教員 植野 真臣 教授

指導教員 宇都 雅輝 准教授

目次

1　はじめに

e テスティングとは，異なる問題で構成されるが，同一精度の測定を実現出来るコンピュータテストのことである [19, 20]．e テスティングを用いることで，同一能力の受験者が異なるテストを受験しても同一得点となる保証がある．そのために，受験者が同一精度で複数回の受験が可能となる [26].

我が国においても情報処理技術者試験「IT パスポート」[29] ，医療系共用試験 [27] 等が e テスティング上で行われている．また，大学入学試験や公務員試験での導入も検討されており，今後益々 e テスティングの需要が高まることが見込まれる．

e テスティングでは一般的に“等質テスト”と呼ばれる，各テストに含まれる出題項目は異なるが，等質なテスト群が生成される．例えば，資格試験等では毎回の難易度が等しくなるように，テストの統計的な性質，得点分布，所要時間が一定でなければならない．これまで，等質テストはテスト管理者の経験と勘により構成されてきたが，e テスティングの普及に伴い，テストを自動構成する手法が数多く提案されている [5, 2, 1, 23, 22, 16].

一般に，e テスティングでは，テストの管理方法としてアイテムバンク方式が用いられる．アイテムバンクとは出題する問題（以降，項目と呼ぶ）の出題分野や統計データ等を格納しているデータベースのことである．このアイテムバンクから所望のテストの性質を満たす項目の組み合わせを計算機により探索することをテストの自動構成と呼ぶ.

この自動構成は組み合わせ最適化問題として解かれる．図 1 は等質テスト自動構成の概念図である．一般に e テスティングの等質テスト構成はアイテムバンクから，互いに受験者得点の予測誤差が等質とな

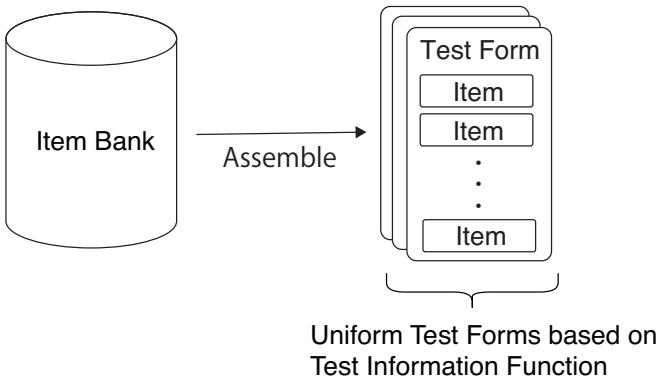

図 1　アイテムバンクからの等質テスト構成

るように異なる項目の組み合わせを列挙する．これにより，同一能力の受検者が異なるテストを受けても同一の得点となることが保証される．先行研究として，Songmuang and Ueno（2010）は最適化問題の解探索手法の一つである Bees Algorithm を用いてテスト構成を行う手法を提案・開発している．この手法は情報処理技術社試験をはじめとして，我が国の国家試験で実際に使用されている [29].

石井ら（2014）は与えられたアイテムバンク・構成条件において，最も多くの等質テストを構成する手法を提案した [7]．この手法はテスト構成問題をグラフ上で定義される最大クリーク問題に帰着させる．具体的には与えられたアイテムバンク・テスト構成条件で構成可能な全てのテストを頂点，二つのテストが等質かつ，共通する項目の数が一定数以下である場合に頂点（テスト）間に辺を引いたグラフから，クリークと呼ばれる任意の二頂点が隣接しているグラフ構造を探索することで等質テスト構成を行う．

この手法は理論的に最大数の等質テスト構成を保証するが，構成可能な全てのテストを頂点とするグラフ構造は，組み合わせ爆発的に大

きくなるため，最大クリークを探索することや，グラフ構造全てをメモリ上に保存することは困難である．そのため，石井ら（2014）はグラフ全域から部分グラフをランダムに抽出し，ここから最大クリーク探索を繰り返すことにより，グラフ全体の最大クリークを近似的に探索する手法を提案した [8]．本手法により，当時の既存研究よりも 10〜100 倍以上多くのテストを構成できた．

しかし，最大クリーク探索はクリークを C とすると，最先端の最大クリーク探索手法 [13, 18] を用いても，$\mathrm{O}(|C|^2)$ の空間計算量を少なくとも必要とするため，（著者らの計算機環境で）最大で 10 万のテストを構成することが限界であった．そこで，石井ら（2017）は探索中のクリーク C の全頂点と隣接する頂点を整数計画法を用いて，逐次的に探索することで，計算に必要な空間計算量を $\mathrm{O}(|C|)$ へ減少させる手法を提案した [10, 30]．これにより，10 万を超える等質テストを構成できるようにした．ただし，整数計画法の時間計算量が $\mathrm{O}(2^n)$（n はアイテムバンクの項目数）と大きく，テスト構成数の改善は僅かなものであった．

整数計画法の計算時間を改善するために，渕本ら（2020）は探索中のクリーク C の全頂点と隣接する頂点を並列探索する手法を提案した [25]．本手法により，最も時間を要している整数計画法による頂点探索を並列化することで，探索時間を大幅に減少できた．さらに，並列化探索で得られた頂点を整数計画法の目的関数の下限値として用いることで，分枝限定法の効果により，探索を高速化した．結果として，最大 1 ヶ月で当時の研究の約 2 倍にあたる約 45 万のテスト生成を実現している．

しかし，この手法を用いても，45 万のテスト生成に約 1 ヶ月も必要

である．例えば，等質テストの導入が検討されている大学入試共通テストでは，年間 50 万人以上が受験している．また，重複して項目を出題することは項目流出のリスクを増大させ，その項目の特性劣化の原因になることが知られている [24]．この問題を避けるために，等質テストは数 10 万〜数 100 万個程度構成することが要求される．この背景の下，より多くのテストを生成できる手法の開発が急務である．

本研究では，より多くのテストを生成するために，ZDD（Zero-suppressed Binary Decision Diagrams; ゼロサプレス型二分決定グラフ）[15] を用いてテスト構成する．ここで，ZDD とは場合分け二分木（Binary Decision Tree）の圧縮表現であり，非循環有向グラフ（Directed Acyclic Graph）により組み合わせ集合を表すデータ構造である．ZDD では場合分け二分木から等価な節点の共有と冗長な節点の削除を行うことで，組み合わせ集合をコンパクトかつ一意なデータ構造で表せる．本手法では，場合分け二分木の深さ i の節点を項目 i の要素とし，各節点を項目 i を選択する 1-枝と選択しない 0-枝で二分する．この場合分け二分木は時間計算量 $O(2^n)$・空間計算量 $O(2^n/n)$ で全ての項目の組み合わせ（テスト）を評価できる．しかし，時間・空間計算量が大きいため，ZDD を用いて場合分け二分木を圧縮する．具体的には，二つの節点に辿り着くまでの全ての組み合わせが等質な場合にそれらを共有することで，場合分け二分木を圧縮する．さらに，生成過程でテスト構成条件を満たさない経路の枝刈りを行う．ただし，場合分け二分木を構築してから圧縮規則を適用する手法はメモリの問題で困難となるため，トップダウン式の ZDD 構築アルゴリズム [12, 11] を用いる．このアルゴリズムでは場合分け二分木を幅優先探索で行い，その過程で冗長な節点の削除と等価な節点の共有を行う．

これにより，圧縮規則を適用しながら，ZDD の構築を行えるため，計算時間やメモリの消費量を抑えられる可能性がある．以上の手続きにより，全ての経路において等質なテストの組み合わせ集合が得られる．その後，それらを統合することで，等質テスト群を構成する．また，項目流出による項目の特性劣化を防ぐため，項目重複数条件を満たすテストの組み合わせ集合を深さ優先探索する．

本論文では提案手法の有効性をシミュレーションデータと実データを用いて示した．その結果，複数の条件において，従来手法が 1 ヶ月を要しても生成できなかった 50 万以上の等質テストを 24 時間以内で生成できた．

2　項目反応理論

等質テストは,「各テストでの受験者得点の予測誤差が等質である」テスト集合として定義される（例えば, [4, 7, 8]）．ここで, 受験者得点の予測誤差はテストの自動構成に関する研究（例えば, [5, 2, 23, 1, 22, 16]）では，項目反応理論（Item Response Theory:IRT）[3, 14] と呼ばれる数理モデルにおけるテスト情報量で評価されている．IRT とは受験者の項目への正答確率をモデル化したものである．これにより，異なる項目から構成されるテストを受けた受験者の能力を同一尺度上で評価できる．

IRT では，項目 $i(=1,...n)$ に対する受験者 $j(=1,...,m)$ の反応 u_{ij} を以下のように表す．

$$u_{i,j} = \begin{cases} 1 & i \text{ 番目の項目に受験者 } j \text{ が正答} \\ 0 & \text{それ以外} \end{cases}$$

本論文では項目反応理論の中で最もよく使われている 2 母数ロジスティックモデル（2-Parameter Logistic Model:2PLM）を用いる．このモデルでは能力値 $\theta_j \in (-\infty, \infty)$ を持つ受験者 j が項目 i に正答する確率 $p_i(\theta_j)$ を以下のように定義する．

$$\begin{aligned} p_i(\theta_j) &\equiv p(u_{ij}=1|\theta_j) \\ &= \frac{1}{1+\exp(-1.7a_i(\theta_j - b_i))} \end{aligned} \tag{1}$$

ただし，$a_i \in [0,\infty], b_i \in [0,\infty]$ はそれぞれ i 番目の項目の識別力パラメータ，困難度パラメータと呼ばれる項目パラメータである．

IRT では項目 i において，式（1）を用いて計算したフィッシャー情報量を項目情報量 $I_i(\theta)$（Item Information）と呼び，以下のように定

表 1　テスト情報量制約の例

$\theta = -2.0$	$\theta = -1.0$	$\theta = 0.0$	$\theta = 1.0$	$\theta = 2.0$
0.0/0.2	0.1/0.3	0.1/0.3	0.1/0.3	0.0/0.2

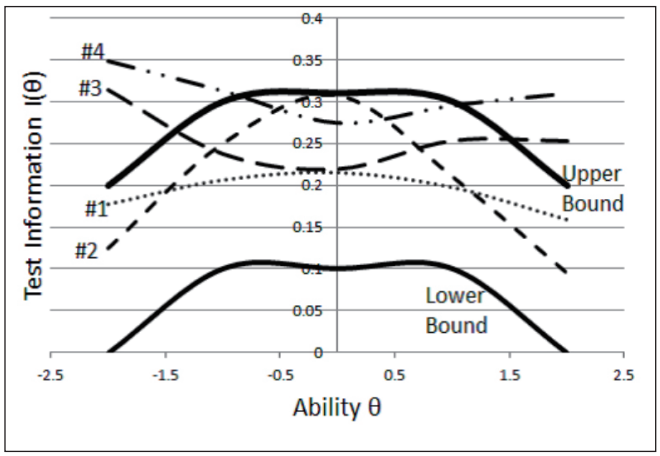

図 2　テスト情報量への上限下限の例

義する．

$$I_i(\theta) = 1.7^2 a_i^2 p_i(\theta)(1 - p_i(\theta)) \tag{2}$$

また，テストに含まれる項目の項目情報量の総和をテスト情報量と呼び，以下のように表す．

$$I(\theta) = \sum_{i \in T} I_i(\theta) \tag{3}$$

ここで，T はテストに含まれる項目の集合である．このテスト情報量の逆数が受験者能力推定値の漸近分散に収束することが知られている [26]．

ただし，テストの自動構成手法では（例えば，[5, 2, 23, 1, 22, 16]）ではテスト情報量における受験者の能力パラメータ θ_k を $\Theta =$

$\{\theta_1, \theta_2, ..., \theta_K\}$ のように幾つかの点でサンプリングし，離散的に扱っている．具体的には, 各点ごとにテスト情報量制約 ($\mathrm{UB}(\theta_k)$, $\mathrm{LB}(\theta_k)$) を設定し，全ての制約を満たすテストを受験者得点の予測誤差が等質であるとする．図 2 は，表 1 に示したテスト情報量制約を与えたときの概念図である．図中の ♯1〜♯4 は構成テストの情報量関数である．♯1, ♯2 は共に制約を満たしており，等質である．一方で，♯3, ♯4 は制約を満たしておらず，等質でない.

3　等質テストの自動構成アルゴリズム

本節では，従来手法の中で最も多くの等質テストを生成可能な PIPMCP 法 [25] 及び，関連する手法を紹介する．

3.1　等質テストのための最大クリーク問題

石井ら（2014）はテスト構成をグラフ上で定義される最大クリーク問題に帰着することで，厳密に最大数の等質テストを構成する手法を提案した [7]．ここで，クリークとは任意の二頂点が隣接している部分グラフである．

また，生成されるテスト候補を以下のグラフ構造とみなし，グラフ構造の中から最大クリークの探索・抽出を行うことで，等質テストを構成する．

（頂点）　テストの構成条件を満たす，与えられたアイテムバンクから構成可能なテスト (以降，テスト候補と呼ぶ) 全てを頂点とする．

（辺）　二つのテスト候補の共通する項目数が一定値以下（以降，項目重複数条件と呼ぶ）の場合，その二つの頂点（テスト）間に辺を引く．

このグラフの任意の頂点はテスト構成条件を満たしている．さらに，クリーク中の任意の二頂点は隣接しており，項目重複数条件を満たす．したがって，このクリーク中の頂点に対応するテストはそれぞれ等質であり，その中でも頂点数が最大のクリークは最大の等質テスト群となる．

結果として等質テスト構成は無向グラフ $G = (V, E)$ を頂点の有限集合 V と辺の集合を E としたとき，次のように定式化できる．

variables $C \subseteq V$

maximize $|C|$

subject to

$\forall v, \forall w \in C, \{v, w\} \in E$

$*$ v, w が $|v \cap w| \leq$ OC(項目重複数条件) を満たすとき辺が引かれる．

この最大クリーク問題を厳密に解くことにより，理論的に最大数を保証した等質テストを出力する．

この手法は厳密に最大数のテストを構成できるアルゴリズムであるが，$O(2^{|V|})$，$O(|V|^2)$ の時間・空間計算量を必要とする．等質テストの場合，グラフの頂点数はアイテムバンクからテストの構成条件を満たすテストの総数となるが，その数はアイテムバンクの項目数 n に対して，組み合わせ爆発的に増加する．ゆえに，現在実施されているような数百 ~ 千以上のアイテムバンクから等質テストの構成を厳密に行うことは困難である．

これらの計算コストを緩和するため，石井ら（2014）は最大クリーク探索を行う近似アルゴリズムを提案した [8]（以降，RndMCP 法と呼ぶ）．最大クリーク法の問題点は等質テスト構成数が増加すると，グラフの探索空間が莫大となることである．そのため，RndMCP 法ではテスト候補グラフ全体から部分グラフをランダムに抽出し，ここから最大クリーク探索を繰り返すことで，グラフ全体の最大クリークを近似的に探索する．これにより，従来手法 [16, 17, 22] と比較して，10 ~1000 倍以上多くのテストを構成できる．

3.2 整数計画法を用いた最大クリークアルゴリズム

RndMCP 法は空間計算量が大きく，10 万個程度のテスト構成が上限であった．そこで, 石井ら (2017) はこの空間計算量を緩和するために，整数計画法を用いた手法を提案した [10, 30]（以降，HybridRBP 法と呼ぶ)．

HybridRBP 法では現在探索中のクリーク C の全頂点と隣接する頂点を以下の整数計画法を用いて，逐次的に探索する．これにより，計算に必要な空間計算量を $\mathrm{O}(|V|)$ へ減少させる．ただし，この探索は $\mathrm{O}(2^n)$ の時間計算量を必要とするため，RndMCP 法で計算機環境のメモリの限界の頂点数を持つグラフから最大クリーク探索を行ってから，整数計画法を用いる方法に切り替えることで，探索効率を改善する [10, 30].

しかし，本手法は整数計画法の探索時間が大きいため，テスト構成数の改善は限定的である.

3.3 等質テスト構成のための整数計画法を用いた最大クリーク問題のアルゴリズム並列化

HybridRBP 法は整数計画法の時間計算量が $\mathrm{O}(2^n)$ と大きく，計算時間を 1 週間用意しても，13 万個程度のテスト構成が限界であった．そこで, 渕本ら (2020) はこの計算時間を改善するために, HybridRBP 法の頂点探索を並列化する手法を提案した [25]（以降，PIPMCP 法と呼ぶ)．

PIPMCP 法では探索効率を改善するために，探索中のクリークの全頂点と隣接する頂点を並列探索する．HybridRBP 法では整数計画法

で求めた頂点を探索中のクリークに追加する度，制約条件を新たに追加しなければならない．そのため，解いた整数計画法の解が次の制約条件を変更し，並列化が困難である．PIPMCP 法では探索中のクリークの全頂点と隣接する頂点集合を候補頂点集合として，逐次的に整数計画法の解を追加し，要素数が一定となるまで繰り返す．この操作は整数計画法の制約条件を変更せずに行えるため，並列化できる．ただし，候補頂点集合中の要素は探索中のクリークの全頂点と隣接しているが，それら自身が互いに隣接している保証は無い．そのため，候補頂点集合の中から最大クリークを抽出し，これを探索中のクリークに追加する．

これらより，HybridRBP 法で最も時間を要している整数計画法で頂点を逐次的に追加する処理を並列化することで，探索時間を大幅に減少できる．さらに，並列化探索で得られた候補頂点集合の要素の目的関数の値を逐次的に次の整数計画法での下限値となり，探索をより高速化できる．

本手法により，最も時間を要している整数計画法による頂点探索を並列化することで，探索時間を大幅に減少できた．さらに，並列化探索で得られた頂点を整数計画法の目的関数の下限値として用いることで，分枝限定法の効果により，探索を高速化できた．本手法により，従来の手法を大きく改善して約 45 万のテストを生成できた．

4　ZDD を用いた等質テスト構成

整数計画法を用いた先行研究 [10, 30, 25] では，一つのテストを生成するための時間計算量が $O(2^n)$ と大きく，テスト構成数の増加は限定的である．実際に，PIPMCP 法は計算時間を 1 ヶ月としても，45 万個程度のテスト構成が限界であった [25].

本研究では，より多くのテストを生成するために，ZDD（Zero-suppressed Binary Decision Diagrams; ゼロサプレス型二分決定グラフ）[15] を用いてテスト構成する．ここで，ZDD とは場合分け二分木（Binary Decision Tree）の圧縮表現である．ZDD では場合分け二分木から等価な節点の共有と冗長な節点の削除を行うことで，組み合わせ集合をコンパクトかつ一意なデータ構造で表せる．本研究では，場合分け二分木の深さ i の節点を項目 i の要素とし，各節点を項目 i を選択する 1-枝と選択しない 0-枝で二分する．この場合分け二分木は時間計算量 $O(2^n)$・空間計算量 $O(2^n/n)$ で全ての項目の組み合わせ（テスト）を評価できるが，時間・空間計算量が大きいため，ZDD を用いて場合分け二分木を圧縮する．具体的には，二つの節点に辿り着くまでの全ての組み合わせが同一な場合にそれらを共有する．さらに，生成過程でテスト構成条件を満たさない経路の枝刈りを行う．以上の手続きにより，全ての経路において等質なテストの組み合わせ集合が得られる．

4.1　ZDD

ZDD[15] とは，図 3-(b) のような，非循環有向グラフ（Directed Acyclic Graph）により，組み合わせ集合を表すデータ構造である．

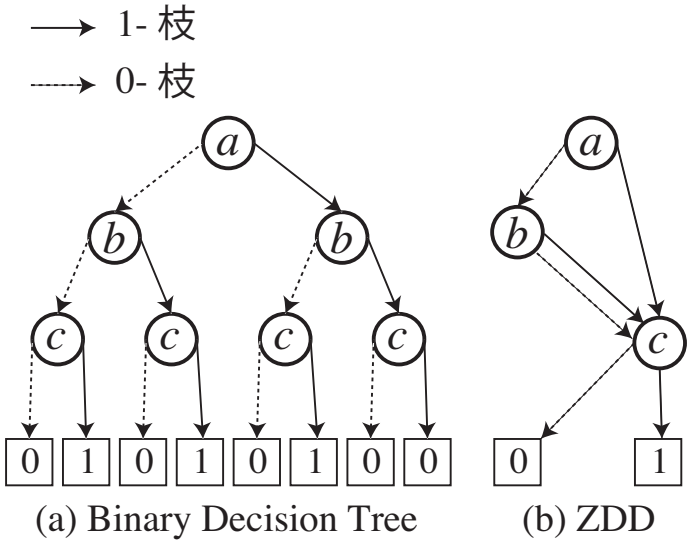

図 3　ZDD と場合分け二分木

ZDD は図 3-(a) のような，場合分け二分木を圧縮することで得られる．場合分け二分木では組み合わせ集合における要素（$\{a, b, c\}$）を節点に割り当てる．また，節点からはその要素が含まれることを表す 1-枝と含まれないことを表す 0-枝が伸びている．さらに，葉には組み合わせ集合が所要の条件を満すという結果を表す 1-終端節点と満たさないという結果を表す 0-終端節点を割り当てる．

この場合分け二分木から ZDD を得るためには，図 3 のように節点に対応する要素（$\{a, b, c\}$）の順序を固定し，以下の圧縮規則を可能な限り適用する．

（冗長節点の削除）　1-枝が 0-終端節点を指している場合に，この節点を取り除く．

（等価な節点の共有）　同じ要素を示す節点が等価（0-枝同士，1-枝同士の行き先が同じ）な場合，節点を共有する．

この手続きにより，組み合わせ集合をコンパクトかつ一意（既約）なデータ構造で表せる．

4.2 ZDD を用いた等質テスト構成

本論文では ZDD を用いた等質テスト構成手法を提案する．はじめに，等質テスト構成のための場合分け二分木を考える．場合分け二分木は深さ i の各節点を項目 i の要素とし，各節点を項目 i を選択する 1-枝と選択しない 0-枝で二分する．この場合分け二分木は時間計算量 $O(2^n)$・空間計算量 $O(2^n/n)$ で全ての項目の組み合わせ（テスト）を評価できる．ただし，時間計算量・空間計算量が大きいため，ZDD を用いてこの場合分け二分木を圧縮する．具体的には項目 i の項目情報量やその節点に到達するまでに選択された項目数を基に，節点の共有やテスト構成条件を満たさない経路を枝刈りする．これにより，等質テスト群を ZDD で表せる．

しかし，場合分け二分木のサイズは項目数に対して，指数的に増加する．そのため，場合分け二分木を構築してから，圧縮する手法はメモリの問題で困難となる．この問題を解決するために，トップダウン式の ZDD 構築アルゴリズム [12, 11] が提案されている．本アルゴリズムでは，場合分け二分木を幅優先探索で行い，その過程で ZDD を圧縮する．これにより，探索時に等価な節点の共有や経路を圧縮できるため，メモリの消費量を抑えられる．

本論文ではこのトップダウン式の ZDD 構築アルゴリズムを用いた等質テスト構成手法を提案する．はじめに，各節点にはその節点までに選択された, 項目数を表す変数 $test\_length$ 及び項目情報量（式 2）の総和を表す変数 $test\_info_k$ を用意する．ここで，項目情報量の計算は

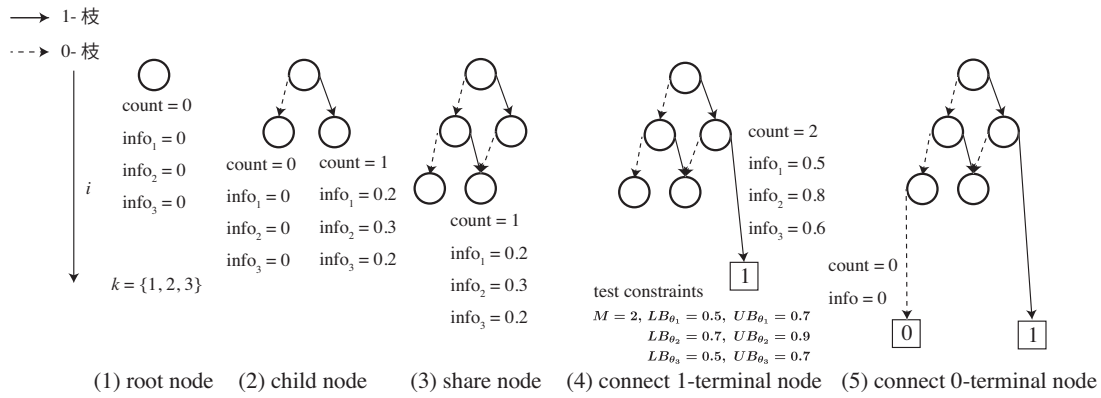

図 4　提案手法の概要

各 $\theta_k (k = 1, 2, \ldots, K)$ について行う．変数 $test\_length$ と $test\_info_k$ を幅優先探索する過程で計算し，節点の共有条件や経路の枝刈りに用いる．また，各変数の値が所望のテスト構成条件（テスト項目数やテスト情報量制約）を満たすとき，1-終端節点に接続する．これにより，等質な項目の組み合わせ（テスト）を列挙できる．

具体的には，図 4 のように 5 つのステップで構成される．

1. 根節点を生成する．根節点ではまだ項目が選択されていないため，変数 $test\_length$ と $test\_info_k (k = 1, 2, \ldots, K)$ の値を 0 とする．
2. 次に，根節点から 0-枝と 1-枝を選択した場合の子節点を生成する．このとき，1-枝を選択した子節点の変数は $test\_length$ の値を一つ加算し，深さ i の項目が θ_k の点で持つ項目情報量を $test\_info_k$ に加算する．一方で，0-枝を選択した子節点は親節点の変数と同値とする．以降，同様の手順で，深さ i の各節点の子節点を順に生成する．
3. 深さ i の二節点において，この変数 $test\_length$ と $test\_info_k$

の値が全て一致するとき，節点を共有可能である．理由は項目数と情報量が同一であり、等質の項目集合とみなすことができるからである．

4. 変数 $test\_length$ と $test\_info_k$ が以下の条件を満たすとき，1-終端節点に接続する．

$$\begin{aligned} test\_length &= M(\text{テスト項目数}) \\ \mathrm{LB}(\theta_k) \leq test\_info_k &\leq \mathrm{UB}(\theta_k) \\ (k &= 1, 2, \ldots, K) \\ &(\text{テスト情報量制約}) \end{aligned}$$

5. 変数 $test\_length$ と $test\_info_k$ が (4) で示した条件を満たさないとき，0-終端節点に接続する．

しかし，アイテムバンクの項目の性質上，変数 $test\_info_k$ の値が完全に一致することはごく稀である．そのため，共有可能な節点数を増加させるために，節点の共有条件を緩和する．具体的には，閾値 I_{th} を用意し，二節点が持つ変数 $test\_info_k$ の差が全て I_{th} 以下となる時，それらを共有する．また，共有後の $test\_info_k$ の値はそれらの節点の平均値とする．ただし，この近似により項目情報量の総和に誤差が生じることにも留意しないといけない．

以上の手続きで，テスト情報量の上限・下限制約を満たす項目の組み合わせが列挙できる．

Algorithm 1 Uniform Tests Assembly

```
procedure TestAssembly(n, M)
    create a new node v_root                                              ▷ root node
    v_root.state.test_length ← 0
    for k ← 1 to K do
        v_root.state.test_info_k ← 0
    end for
    N_1 ← {v_root}                                          ▷ N_i is node sets of depth i
    for i ← 2 to n do
        N_i ← ∅
    end for
    N_{n+1} ← {0-terminal node, 1-terminal node}
    for i ← 1 to n do
        for each v ∈ N_i do
            for each x ∈ {0, 1} do                                        ▷ 0-arc, 1-arc
                {i', state'} ← Child(i, M, v.state, x)
                ▷ i' is depth of child node. state' is test_length and test_info_k of child node.
                v' ←create a new node                                     ▷ child node
                if {i', state'} is {n + 1, 0} then
                    v' ← 0-terminal node
                else if {i', state'} is {n + 1, 1} then
                    v' ← 1-terminal node
                else
                    v'.state ← state'
                    share_node ← False
                    for each w ∈ N_{i+1} do
                        if v'.state.test_length = w.state.length then
                            for k ← 1 to K do
                                if I_th ≤ |v'.state.test_info_k − w.state.test_info_k| then
                                    next w
                                end if
                            end for
                            UpdateState(v', w)
                            v' ← w                                        ▷ share node
                            share_node ← True
                            break
                        end if
                    end for each
                    if share_node is False then
                        N_{i+1} ← N_{i+1} ∪ v'
                    end if
                end if
                v.child[x]← v'
            end for each
        end for each
    end for
    return Reduce(v_root)
end procedure
procedure Child(i, M, state, x)
    if x = 1 then
        state'.test_length ← state.test_length + 1
        for k ← 1 to K do
            state'.test_info_k ← state.test_info_k + I_i(θ_k)             ▷ I_i(θ_k) in eq(2)
        end for
    end if
    if state'.test_length = M then                                        ▷ test length constraint
        for k ← 1 to K do
            if not LB_{θ_k} < state'.test_info_k < UB_{θ_k} then          ▷ test information constraint
                return {n + 1, 0}                                         ▷ 0-terminal node
            end if
        end for
        return {n + 1, 1}                                                 ▷ 1-terminal node
    end if
    if state'.test_length + n − i < M then
        for k ← 1 to K do
            if UB_{θ_k} < state'.test_info_k then
                return {n + 1, 0}                                         ▷ 0-terminal node
            end if
        end for
    end if
    return {i + 1, state'}
end procedure
procedure UpdateState(v', w)
    for k ← 1 to K do
        w.state.test_info_k ← (v'.state.test_info_k + w.state.test_info_k)/2
    end for
end procedure
```

具体的なアルゴリズムを Algorithm1 に示す.

はじめに，根節点を示す v_{root} を生成し，その変数を $test\_length = 0, test\_info_k = 0 (k = 1, 2, \ldots, K)$ で初期化する（2〜6 行目）．また，深さ i の各節点を格納する配列 N_i を用意する．ここで，生成した根節点 v_{root} は N_1 に追加し，$N_i (i = 2, 3, \ldots, n)$ の要素は $\emptyset$ で初期化する（7〜10 行目）．また，N_{n+1} には 0-終端節点と 1-終端節点を示す節点が格納されている（11 行目）.

以降，深さ i の各節点 v を節点集合 N_i から取り出し，1-枝と 0-枝を選択した場合の子節点 v' を順に生成する（12〜45 行目）．また，子節点の変数 $test\_length$ 及び $test\_info_k$ の計算はプロシージャ CHILD を用いて計算する（48〜71 行目）．その計算結果から，節点の共有や子節点または終端節点への接続を行う（17〜42 行目）．ここで，節点の共有を行う場合はプロシージャ UPDATESTATE を用いて，変数 $test\_info_k$ の値を二節点の平均値で更新する（72〜76 行目）.

トップダウン構築式で生成された ZDD は深さ i の節点共有のみ行うため，さらに圧縮規則が適用できる可能性がある．そのため，生成された ZDD に対して圧縮規則を適用する（32 行目）.

4.3 重複項目を考慮した等質テスト ZDD

4.2 で提案した ZDD を用いた等質テスト構成手法は従来手法 [8, 10] で導入されている重複項目数条件（任意の二つのテストにおける共通項目数が一定値以下）が考慮されていない．重複して項目を出題することは項目流出のリスクを増大させ，その項目の特性劣化の原因になることが知られている [24]．これは 4.2 の手法の場合，任意の経路間で共通項目数が一定値以下となる必要がある．このため，4.2 の手法を

用いて生成した ZDD から，重複項目数条件を満たす経路を探索する手法を提案する．

ただし，ZDD を用いた等質テスト構成手法は節点を共有する際に，項目情報量を平均値で近似するため誤差が生じる．このため，テスト情報量制約を満たさない経路が存在する可能性がある．この問題を解決するために，重複項目数条件を満たす経路を探索する際に，テスト情報量制約を再評価することで，制約を満たさない経路を除外する．

Algorithm2 に詳細を示す．また，アルゴリズムの引数には任意の項目重複数条件 (OC) を与える．はじめに, 項目重複数条件とテスト情報量制約を満たす等質テストの組み合わせ集合を格納する変数 $test\_set$ を用意する（2 行目）．また，アルゴリズムは設定された制限時間の範囲で探索する（ 3 〜20 行目）．その後，4.2 の手法を用いて生成した ZDD から逐次的に経路を取り出し，項目重複数条件とテスト情報量制約をチェックする（4〜19 行目）．ここで，経路を取り出す方法は様々考えられるが，提案手法では一様ランダムサンプリングにより，経路を 1 つずつ取り出す．次に，取り出した経路を（$\theta_k(k = 1, 2, \ldots, K)$）について（5〜12 行目），平均近似しない厳密なテスト情報量（式 (3)）を計算する（6 行目）．このとき，テスト情報量制約を満たさなければ，次の経路を取り出す（8〜11 行目）．さらに，集合 $test\_set$ の全テストと項目重複数条件を満たす場合，集合 $test\_set$ にその経路を追加する（13〜18 行目）．このとき，重複項目数条件を満たさないテストが 1 つでも存在した段階で次の経路を取り出す（15 行目）．

Algorithm 2 ZDD Overlap(Overlap)

```
1: procedure ZDD Overlap(OC)
2:     test_set ← ∅                                          ▷ uniform test set
3:     while within a time limitation do
4:         for each test (path) ∈ all paths do
5:             for each k ∈ K do
6:                 info_θk ← CalculateInfo(test, θ_k)
7:                                                           ▷ eq(3)
8:                 if LB_θk < info_θk < UB_θk then
9:                                                  ▷ test information constraint
10:                    Next path
11:                end if
12:            end for
13:            for each t ∈ test_set do
14:                if OC < |test ∩ t| then                   ▷ overlap constraint
15:                    Next path
16:                end if
17:            end for
18:            test_set ← test_set ∪ test
19:        end for
20:    end while
21:    return test_set
22: end procedure
```

5　評価実験

5.1　節点共有条件による節点の圧縮効果とテスト構成数

提案手法では，情報量が一定値 I_{th} 以下の場合に節点の共有を行う．そのため，I_{th} は ZDD の節点数やテスト構成数，計算時間に影響を与える．この影響を評価するために，大規模アイテムバンクを用いた提案手法の比較実験を行った．具体的には，提案手法のパラメータ I_{th} の値を 0.10 ～ 0.25 まで 0.01 ずつ変化させ，影響を比較する．

実験には三つのシュミレーションアイテムバンクおよび実データアイテムバンクを用いた．シュミレーションアイテムバンクは 500,1000,2000 の項目を持ち，各項目の識別力パラメータ a を $\log_2 a \sim N(0, 1^2)$，困難度パラメータ b を $b \sim N(0, 1^2)$ として発生させた．ま

表 2　実アイテムバンクの詳細

Item Bank Size	Parameter a			Parameter b		
	Range	Mean	SD	Range	Mean	SD
87	0.15 ～ 0.67	0.35	0.134	−2.09 ～ 4.55	0.73	1.625
93	0.19 ～ 0.69	0.43	0.122	−3.92 ～ 3.61	−0.79	1.196
104	0.13 ～ 1.10	0.59	0.213	−0.18 ～ 4.55	1.50	1.188
141	0.24 ～ 1.09	0.64	0.155	−1.41 ～ 3.91	0.60	0.855
158	0.15 ～ 3.08	0.44	0.255	−4.00 ～ 4.00	−1.12	1.434
175	0.12 ～ 0.93	0.39	0.139	−2.93 ～ 3.12	−0.25	1.113
220	0.16 ～ 0.92	0.46	0.155	−4.00 ～ 2.82	−1.28	1.098
Total: 978	0.12 ～ 3.08	0.46	0.198	−4.00 ～ 4.55	−0.22	1.572

表 3　テスト構成のためのテスト情報量条件の下限値/上限値

$\theta = -2.0$	$\theta = -1.0$	$\theta = 0.0$	$\theta = 1.0$	$\theta = 2.0$
2.0/2.4	3.2/3.4	3.2/3.4	3.2/3.4	2.0/2.4

た，実データアイテムバンクの詳細は表 2 のとおりである．

テストの構成条件は前述したアイテムバンクから，表 3 のテスト情報量条件を満たす 25 項目のテスト構成とした．本条件は実際に運用された e テスティングにおけるテスト構成条件であり，実際に運用されている規模のテスト構成を模倣している．なお，本論文での計算機環境は Ubuntu 18.04.2 を OS とする計算機（CPU:Intel Core i9-9900X 3.50 GHz，RAM:128GB）である．

結果を表 4 に示す．No.tests はテスト構成数，Vertices は ZDD の節点数，Time[min] は計算時間である．表 4 より，I_{th} の値が小さいほど，テスト構成数は増加傾向にあるが，共有節点数が減少するため，メモリオーバーを引き起こす．例えば，Item Bank Size = 978 の場合，$I_{th} = 0.11$ 以下に設定するとメモリオーバーを引き起こした．一方で，I_{th} の値を大きくすると，節点の共有条件が緩和されるため，ZDD の節点数及び計算時間は減少する．

また，提案手法は節点共有時に項目情報量を平均値で近似する．そのため，I_{th} の値に応じて項目情報量の総和に誤差が生じる．この影響を調べるため，節点共有時における二節点の項目情報量の差を全て記録し，その平均値と標準誤差を求めた．ただし，メモリオーバーを引き起こした条件については記録していない．表 5 は各アイテムバンクにおいて，I_{th} の値を変化させて平均値と標準誤差が最も小さくなったとき（上段）とテスト構成数が最大となったとき（下段）の値である．表 5 より，アイテムバンクの項目数が増加するにつれて，僅かに項目情報量の誤差の平均値や偏りも増加する傾向がある．また，テスト構成数が最大の場合の誤差は平均値と標準誤差が最小の場合と比較して僅かに増加する．しかし，提案手法は Algorithm2 を用いて，厳密な

テスト情報量を再計算するため，本論文ではテスト構成数が最も大きくなる I_{th} の値を採用する．

表 4　I_{th} によるテスト構成数及び節点数と計算時間の変化

	Item Bank Size = 500			Item Bank Size = 1000			Item Bank Size = 2000			Item Bank Size = 978		
I_{th}	No.tests	Vertices	Time [min]	No.tests	Vertices	Time [min]	No.tests	Vertices	Time [min]	No.tests	Vertices	Time [min]
0.11 以下	Memory Over			Memory Over			Memory Over			Memory Over		
0.12	Memory Over			Memory Over			Memory Over			6.6×10^{45}	6.3×10^{7}	767
0.13	Memory Over			Memory Over			Memory Over			4.8×10^{45}	4.0×10^{7}	315
0.14	$\mathbf{2.6 \times 10^{38}}$	2.5×10^{7}	218	Memory Over			Memory Over			4.5×10^{45}	2.5×10^{7}	124
0.15	1.6×10^{38}	1.7×10^{7}	94	8.8×10^{45}	5.1×10^{7}	377	Memory Over			1.9×10^{45}	1.8×10^{7}	61
0.16	7.2×10^{37}	1.2×10^{7}	51	8.6×10^{45}	3.7×10^{7}	208	Memory Over			$\mathbf{1.4 \times 10^{46}}$	1.2×10^{7}	27
0.17	4.8×10^{37}	9.2×10^{6}	28	$\mathbf{1.0 \times 10^{46}}$	2.4×10^{7}	86	3.6×10^{52}	6.7×10^{7}	298	2.6×10^{44}	8.1×10^{6}	13
0.18	3.2×10^{37}	6.5×10^{6}	14	4.7×10^{45}	1.7×10^{7}	46	4.2×10^{52}	4.4×10^{7}	131	1.9×10^{45}	6.7×10^{6}	8
0.19	3.2×10^{36}	5.0×10^{6}	8	7.9×10^{45}	1.2×10^{7}	23	5.8×10^{52}	3.1×10^{7}	66	9.7×10^{44}	4.4×10^{6}	4
0.20	1.3×10^{37}	3.8×10^{6}	5	7.0×10^{45}	1.0×10^{7}	17	$\mathbf{5.2 \times 10^{53}}$	2.5×10^{7}	46	8.5×10^{42}	3.7×10^{6}	2
0.21	3.7×10^{37}	2.9×10^{6}	3	1.6×10^{45}	7.3×10^{6}	8	9.2×10^{52}	1.7×10^{7}	22	2.3×10^{45}	2.5×10^{6}	1
0.22	5.7×10^{36}	2.4×10^{6}	2	7.7×10^{44}	5.6×10^{6}	5	1.7×10^{53}	1.3×10^{7}	12	3.1×10^{44}	2.0×10^{6}	0
0.23	1.1×10^{36}	1.9×10^{6}	1	4.9×10^{44}	4.5×10^{6}	3	6.4×10^{52}	1.0×10^{7}	8	7.6×10^{44}	1.5×10^{6}	0
0.24	1.9×10^{36}	1.5×10^{6}	0	6.2×10^{45}	3.3×10^{6}	2	1.8×10^{53}	9.0×10^{6}	6	2.5×10^{44}	1.2×10^{6}	0
0.25	1.3×10^{33}	$\mathbf{1.2 \times 10^{6}}$	0	4.7×10^{45}	$\mathbf{2.7 \times 10^{6}}$	1	4.7×10^{53}	$\mathbf{6.9 \times 10^{6}}$	3	0	0	0

表 5　節点共有時における項目情報量の誤差

Item Bank		$\theta = -2.0$		$\theta = -1.0$		$\theta = 0.0$		$\theta = 1.0$		$\theta = 2.0$	
Size	I_{th}	mean	std	mean	std	mean	std	mean	std	mean	std
	0.14	0.056	0.040	0.060	0.040	0.060	0.040	0.060	0.040	0.056	0.040
500	0.14	0.056	0.040	0.060	0.040	0.060	0.040	0.060	0.040	0.056	0.040
	0.15	0.058	0.042	0.065	0.043	0.066	0.043	0.065	0.043	0.058	0.042
1000	0.17	0.064	0.047	0.072	0.048	0.073	0.049	0.072	0.049	0.063	0.047
	0.17	0.064	0.048	0.074	0.050	0.077	0.051	0.077	0.051	0.068	0.050
2000	0.20	0.068	0.054	0.081	0.056	0.083	0.057	0.082	0.056	0.070	0.054
	0.12	0.048	0.033	0.050	0.033	0.048	0.033	0.042	0.031	0.032	0.028
978	0.16	0.054	0.042	0.058	0.042	0.056	0.042	0.046	0.038	0.034	0.033

5.2 従来手法との比較実験

提案手法の有効性を示すため, 従来手法（RndMCP 法 [8], PIPMCP 法 [25]）とテスト構成数を比較した．本実験のテスト構成条件は 5.1 と同一である．また，項目重複数条件（OC）は各アイテムバンク毎に $0, 5, 10$ と変化させ，複数の条件によって評価する．

各手法の計算時間は最大 24hr とし，RndMCP 法のパラメータは $L_1 = 100,000$，$L_2 = 3$hour，CT = 24hour，PIPMCP 法のパラメータは $D_1 = 100$，$S_{\mathrm{UB}} = 100$，$P = 10$，CT = 24hour とした．また，提案手法のパラメータは 5.1 の実験でテスト構成数が最も大きくなった I_{th} の値を採用した．

なお，PIPMCP 法 [30] の整数計画法の探索は CPLEX[6] を用い，LP 緩和問題との解ギャップが e^{-4} 以下で計算を打ち切った（デフォルトのオプション）．

各手法によるテスト構成数を表 6 に示す．提案手法は OC の値が大

表 6　大規模アイテムバンクにおけるテスト構成数の比較

Item Bank Size	OC	RndMCP	PIPMCP	Proposal
500	0	10	**17**	7
	5	4380	**14331**	3166
	10	99983	127149	**583008**
1000	0	17	**34**	17
	5	46305	**97492**	35304
	10	100000	131300	**641619**
2000	0	32	**70**	18
	5	96876	129257	**163240**
	10	100000	140700	**651232**
978	0	16	**35**	10
	5	40814	73693	**80102**
	10	100000	124200	**158813**

きくなるほど，従来手法より多くのテストを生成できた．特に，従来手法が 1 ヶ月を要しても生成できなかった 50 万以上の等質テスト構成を 24 時間以内で行い，計算時間を改善できた．これは年間 50 万人以上が受験している大学入試共通テストでも実用可能なテスト構成数及び計算時間である．

提案手法は ZDD により最大で 5.2×10^{53} の情報量制約を満たすテストを 1 時間程度で生成できる．ゆえに，ここから重複項目数条件を満たすテストを探索することで，従来手法よりも多くの等質テスト群を生成できる．ただし，提案手法のテスト構成数が 5 万未満となる条件では，PIPMCP 法が最も多くのテストを生成できた．特に，OC=0 の場合，全ての条件において，PIPMCP 法が最も多くのテストを生成できた．これは OC=0 の場合，制約が厳しく PIPMCP では厳密解を求められるが，提案手法では項目情報量を平均値で近似するため，誤差が生じたことが原因と考えられる．前述したように，この誤差が大きいと，平均値で近似したことにより，本来は情報量制約を満たすはずの経路が枝刈りされた可能性が高い．制約が厳しい条件においては

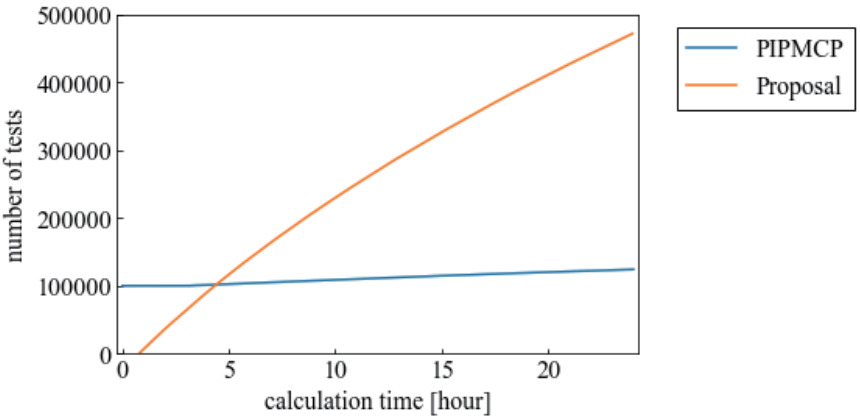

図 5　提案手法と PIPMCP 法のテスト構成数の時間変化

このような経路が枝刈りされることで，テスト構成数の減少に繋がったと考えられる．

図 5 はテスト構成数の時間変化をプロットしたものである（Item Bank Size = 2000, OC = 10）．具体的には，横軸が計算時間 [hour]，縦軸がテスト構成数である．図 5 より，提案手法は時間経過とともに，従来手法とテスト構成数の差が広がる．また，24 時間以内ではテスト構成数が収束しなかったため，計算時間を増やすことで，更にテスト構成数の差が広がる可能性がある．

6　むすび

本論文ではｅテスティングにおける等質テストのための ZDD を用いた等質テストの自動構成アルゴリズムを提案した．本手法は ZDD を用いて，等質テストの組み合わせ集合をコンパクトなデータ構造で表現することで従来手法よりも多くのテストを生成できた．具体的には，場合分け二分木の深さ i の節点を項目 i の要素とし，各節点を項目 i を選択する 1-枝と選択しない 0-枝で二分する．この場合分け二分木は時間計算量 $O(2^n)$・空間計算量 $O(2^n/n)$ で全ての項目の組み合わせ（テスト）を評価できるが，時間・空間計算量が大きいため，ZDD を用いて場合分け二分木を圧縮する．圧縮は二つの節点に辿り着くまでの全ての組み合わせが同一な場合にそれらを共有する．さらに，生成過程でテスト構成条件を満たさない経路の枝刈りを行う．これにより，計算時間や節点数を削減し，多くの等質テストを生成できた．ただし，重複して項目を出題することは項目流出のリスクを増大させ，その項目の特性劣化の原因となるため，構築した ZDD から先行研究 [7, 8, 30, 10, 25] と同様に，項目重複数条件を満たす等質テスト群を ZDD から深さ優先探索により求めた．その結果，従来手法が 1 ヶ月を要しても生成できなかった 50 万以上の等質テスト構成を 24 時間以内で完了できた．

また，本手法の問題と今後の課題には以下がある．本手法での構成テスト群には，項目の露出（出題回数）に偏りが生じる．例えば，露出が多い項目は受験者間で共有されやすく，経年劣化につながり，その項目の信頼性が失われやすい [24]．そのため，この露出を制御できる手法を今後検討する．例えば，Ishii and Ueno（2015）では出題回数が最大となる項目がテスト群に占める割合を軽減する手法が提案さ

れている [9].

さらに，等質テストは適応型テストに用いることで，テストの長さや項目の露出数を軽減することが知られている [21, 28]．適応型テストとは，受験者の能力を逐次的に推定し，その能力に応じて測定精度が最も高い項目を出題することで受験時間や項目数を軽減できるコンピュータ・テスティングの出題形式である．このような，実用上の課題についても検討する．

謝辞

本論文を作成するにあたり，指導教員の植野真臣教授から，丁寧かつ熱心なご指導を賜りました．ここに感謝の意を表します．また，研究についてご助言をいただきました宇都雅輝准教授，研究に関する議論や論文執筆についてご指摘いただきました先輩方，研究室の皆様に感謝いたします．

参考文献

[1] Ronald D Armstrong, Douglas H Jones, and Charles S Kunce. Irt test assembly using network-flow programming. *Applied Psychological Measurement*, Vol. 22, No. 3, pp. 237–247, 1998.

[2] Ronald D Armstrong, Douglas H Jones, and Zhaobo Wang. Automated parallel test construction using classical test theory. *Journal of Educational Statistics*, Vol. 19, No. 1, pp. 73–90, 1994.

[3] Frank B Baker and Seock Ho Kim. *Item response theory: Parameter estimation techniques*. CRC Press, 2004.

[4] Dmitry I Belov and Ronald D Armstrong. A constraint programming approach to extract the maximum number of non-overlapping test forms. *Computational Optimization and Applications*, Vol. 33, No. 2-3, pp. 319–332, 2006.

[5] Ellen Boekkooi-Timminga. The construction of parallel tests from irt-based item banks. *Journal of Educational Statistics*, Vol. 15, No. 2, pp. 129–145, 1990.

[6] IBM. Ilog cplex optimization studio cplex user's manual 12.9, 2019.

[7] Takatoshi Ishii, Pokpong Songmuang, and Maomi Ueno. Maximum clique algorithm for uniform test forms assembly. In *International Conference on Artificial Intelligence in Education*, pp. 451–462. Springer, 2013.

[8] Takatoshi Ishii, Pokpong Songmuang, and Maomi Ueno. Maximum clique algorithm and its approximation for uniform test

form assembly. *IEEE Transactions on Learning Technologies*, Vol. 7, No. 1, pp. 83–95, 2014.

[9] Takatoshi Ishii and Maomi Ueno. Clique algorithm to minimize item exposure for uniform test forms assembly. In *International Conference on Artificial Intelligence in Education*, pp. 638–641. Springer, 2015.

[10] Takatoshi Ishii and Maomi Ueno. Algorithm for uniform test assembly using a maximum clique problem and integer programming. In *International Conference on Artificial Intelligence in Education*, pp. 102–112. Springer, 2017.

[11] Hiroaki Iwashita and Shin-ichi Minato. *Efficient top-down ZDD construction techniques using recursive specifications*, 2013.

[12] Donald Ervin Knuth. The art of computer programming: Bitwise tricks & techniques. *Binary Decision Diagrams*, Vol. 4, , 2009.

[13] Chu Min Li, Hua Jiang, and Felip Many‘a. On minimization of the number of branches in branch-and-bound algorithms for the maximum clique problem. *Computers & Operations Research*, Vol. 84, pp. 1–15, 2017.

[14] Frederic M Lord and Melvin R Novick. *Statistical theories of mental test scores*. IAP, 2008.

[15] Shin-ichi Minato. Zero-suppressed bdds for set manipulation in combinatorial problems. In *Proceedings of the 30th International Design Automation Conference*, pp. 272–277, 1993.

[16] Pokpong Songmuang and Maomi Ueno. Bees algorithm for construction of multiple test forms in e-testing. *IEEE Transactions on Learning Technologies*, Vol. 4, No. 3, pp. 209–221, 2010.

[17] Koun Tem Sun, Yu Jen Chen, Shu Yen Tsai, and Chien Fen Cheng. Creating irt-based parallel test forms using the genetic algorithm method. *Applied measurement in education*, Vol. 21, No. 2, pp. 141–161, 2008.

[18] Etsuji Tomita, Sora Matsuzaki, Atsuki Nagao, Hiro Ito, and Mitsuo Wakatsuki. A much faster algorithm for finding a maximum clique with computational experiments. *Journal of Information Processing*, Vol. 25, pp. 667–677, 2017.

[19] Maomi Ueno. Ai based e-testing as a common yardstick for measuring human abilities. In *2021 18th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 1–5. IEEE, 2021.

[20] Maomi Ueno, Kazuma Fuchimoto, and Emiko Tsutsumi. E-testing from artificial intelligence approach. *Behaviormetrika*, Vol. 48, No. 2, pp. 409–424, 2021.

[21] Maomi Ueno and Yoshimitsu Miyazawa. Uniform adaptive testing using maximum clique algorithm. In *International Conference on Artificial Intelligence in Education*, pp. 482–493. Springer, 2019.

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

[23] Wim J van der Linden and Jos J Adema. Simultaneous assembly of multiple test forms. *Journal of educational measurement*, Vol. 35, No. 3, pp. 185–198, 1998.

[24] Howard Wainer. Cats: Whither and whence. *Psicologica*, Vol. 21, No. 1, pp. 121–133, 2000.

[25] 渕本壱真, 植野真臣. 等質テスト構成における整数計画法を用いた最大クリーク探索の並列化. 電子情報通信学会論文誌 D, Vol. 103, No. 12, pp. 881–893, 2020.

[26] 植野真臣, 永岡慶三. e テスティング. 培風館, 2009.

[27] 仁田善雄, 齋藤宣彦, 後藤英司, 高木康, 石田達樹, 江藤一洋. 医療系大学間共用試験における e テスティング. 日本テスト学会 第 12 回大会 発表論文抄録集, pp. 58–59, 2014.

[28] 宮澤芳光, 宇都雅輝, 石井隆稔, 植野真臣. 測定精度の偏り軽減のための等質適応型テストの提案. 電子情報通信学会論文誌 D, Vol. 101, No. 6, pp. 909–920, 2018.

[29] 谷澤明紀, 本多康弘. 情報処理技術者試験における e テスティング. 日本テスト学会 第 12 回大会 発表論文抄録集, pp. 54–57, 2014.

[30] 石井隆稔, 赤倉貴子, 植野真臣. 複数等質テスト構成における整数計画問題を用いた最大クリーク探索の近似法. 電子情報通信学会論文誌 D, Vol. 100, No. 1, pp. 47–59, 2017.