論　文

# 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

渕本　壱真[†a)]　　植野　真臣[†]

# Parallel Maximum Clique Algorithm Using Integer Programming for Uniform Test Forms Assembly

Kazuma FUCHIMOTO[†a)] and Maomi UENO[†]

**あらまし**　本論文では，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の並列化探索手法を提案する．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．石井ら（2017）は等質テスト構成を最大クリーク問題と整数計画法により，従来手法より多くのテストを構成できる手法を提案した．本論文では，石井ら（2017）の手法で最も時間を要する整数計画法による逐次的頂点追加処理を並列化することを考える．具体的には，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並列化して繰り返し，この要素の最大クリークを追加することで探索を高速化できる．この提案手法の有効性は最大で従来手法で生成されるテスト数が 194575 個であったのに対し，438950 個にテスト生成数を更新した．

**キーワード**　e テスティング，項目反応理論，等質テスト自動構成，最大クリーク問題，整数計画法

## 1. まえがき

e テスティングとは，異なる問題で構成されるテストが，同一精度の測定を実現できるコンピュータテストのことである．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．他にも様々な利点をもつことが知られている [1].

我が国においても情報処理技術者試験「IT パスポート」[2]，医療系共用試験 [3] 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 [4]〜[9].

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.

この自動構成は数学的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

[†] 電気通信大学大学院情報システム学研究科，調布市
Graduate School of Information Systems, The University of Electro-Communications, Chofu-shi, 182–8585 Japan
a) E-mail: fuchimoto@ai.lab.uec.ac.jp
DOI:10.14923/transinfj.2020JDP7004

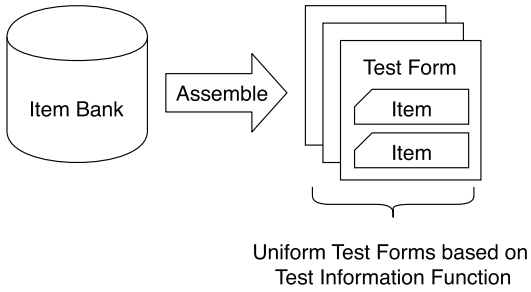


図 1　アイテムバンクからの等質テスト構成
Fig. 1　Uniform test forms assembly from item bank.

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 [2].

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 [10]．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，組み合わせが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 [11]．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 [12], [13] を用いても，$O(|C|^2)$ 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 $O(|C|)$ へ減少させる手法を提案した [14], [15]．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，この手法でも等質テストの構成数が増加するにつれて，整数計画法の探索時間が増加するため，未だ十分な数の等質テストを生成できていない．例えば，既に等質テストが導入されている情報処理技術者試験では年間 20 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 [16]．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．

本論文では探索中のクリークの全頂点と隣接する頂点を並列に探索する手法を提案し，生成されるテスト数を増加させる．石井ら（2017）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．提案手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ない．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．

この提案手法により従来手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．更に，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．具体的には最大で従来手法で生成されるテスト数が 194575 個であったのに対し，提案手法では 438950 個にテスト生成数を更新できた．

## 2. 項目反応理論

一般的に，等質テストは，以下の条件を満たすテスト集合として定義する（例えば，[10], [11], [17]）．

（1） それぞれのテストでの受験者得点の予測誤差が等質である．

（2） それぞれのテスト間の項目重複数が一定値以下である．（以降，項目重複数条件と呼ぶ）

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，[4]〜[9]）では，項目反応理論（Item Response Theory: IRT）[18], [19] 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 $\theta_j \in (-\infty, \infty)$ をもつ受験者 $j$ 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．

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

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 $I_i(\theta)$ と呼び，以下のように定義する．

$$I_i(\theta) = 1.7^2 a_i^2 p_i(\theta)(1 - p_i(\theta)) \quad (2)$$

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．

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

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 [1].

ただし，テストの自動構成手法では（例えば，[4]〜[9]）ではテスト情報量における受験者の能力パラメータ $\theta_i$ を $\Theta = \{\theta_1, ..., \theta_k, ..., \theta_K\}$ 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．

## 3. 等質テストの自動構成アルゴリズム

本章では提案手法と関連がある手法を紹介する．

### 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 [10]．ここで，クリークとは任意の二頂点が隣接しているグラフ構造である．

本手法では能力パラメータ $\Theta = \{\theta_1, ..., \theta_k, ..., \theta_K\}$ をサンプリングし，各点ごとにテスト情報量の上下限制約（$\mathrm{UB}(\theta_k)$, $\mathrm{LB}(\theta_k)$）を設定し，全ての上下限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．

例えば，図 2 は，表 1 に示したテスト情報量の上下限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 は共にこの制約の上下限を満たしており，等質である．一方で，♯3, ♯4 は上下限を満たしておらず，等質でない．

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

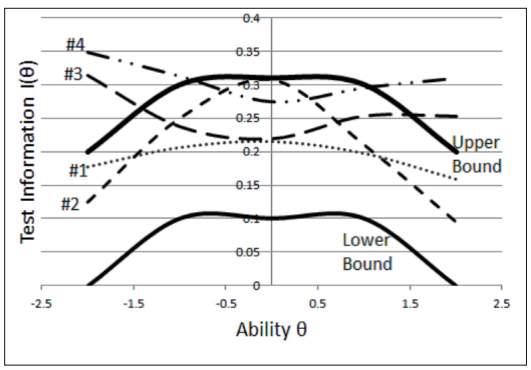


図 2 テスト情報量への上限下限の例
Fig. 2 An image of upper and lower bound for test information.

表 1 テスト情報量の上下限の例
Table 1 An example of upper and lower bound for test information.

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

と呼ぶ) 全てを頂点とする．

辺：二つのテスト候補が項目重複条件を満たす場合，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．更に，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**

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

∗ where,$\{v, w\} \in E$ means the vertices pair of $v, w$ is connected ($|v \cap w| \geq$ (upper bound of the number of overlapping items)).

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．

例えば，図 3 はテスト候補グラフから最大クリークを抽出して等質テストを構成する例を示している．図 3 中には六つの頂点（テスト）と項目重複数条件を満たす辺が 9 本ある．例えば，T5 と T6 はそれぞれテスト構成条件を満たすテスト候補で，項目重複数条件を満たすため，辺で結ばれている．このグラフ中の最大クリークは{T1，T2，T3，T4}であり，これらのテストを抽出すると，与えられたアイテムバンクから構成できる最大数の等質テストとなる．

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，組み合わせが爆発的に増加する．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．

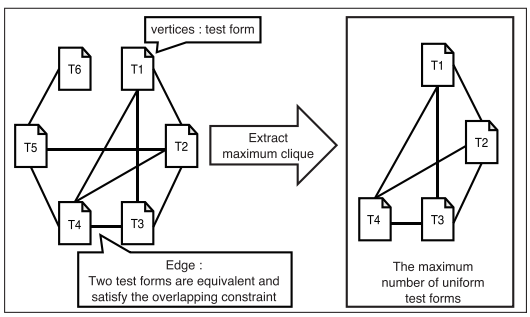


図 3　等質テスト構成のためのグラフ構造
Fig. 3　Graph structure for uniform test assembly.

### 3.2 乱　択　法

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 [11]（以降，RndMCP 法と呼ぶ）．この手法はそれ以前の手法 [4], [5], [20] と比較して，10〜100 倍以上多くのテストを構成できる．

**3.1** で紹介した手法 [10] 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．

具体的には Algorithm 1 により，テスト構成を行う．

Step1〜2 ではテスト構成条件の項目重複数条件以外を満たす $L_1$ 個の頂点（テスト）をもつテスト候補グラフの部分グラフをランダムに抽出する．ただし，$L_1$ はチューニングパラメータであり，メモリ上に保持できる頂点数の上限を計算機環境に合わせて設定する．Step3 では抽出した部分グラフの最大クリーク探索を

**Algorithm 1** 乱択法

**Require:** アイテムバンク，テスト構成条件
**Ensure:** 等質テスト群
1: **procedure** RndMCP($L_1, L_2, CT$)
2: $C \coloneqq \emptyset, C_{max} \coloneqq \emptyset$
3: $ST \coloneqq current\ time$
4: **while** $(current\ time - ST) < CT$ **do**
5: /* Step1 */
6: $V \coloneqq L_1$個のテストをランダム生成
7: ▷ 項目重複数条件以外を満たす $L_1$個のテスト
8: /* Step2 */
9: $G = (V, E)$ グラフ構築
10: ▷ 二頂点が重複項目数条件を満たす場合，辺を引く
11: /* Step3 */
12: $C \coloneqq$ MCP($G, L_2$)
13: ▷ $G$ の最大クリークを時間 $L_2$だけ探索
14: **if** $|C_{max}| < |C|$ **then**
15: $C_{max} \coloneqq C$
16: **end if**
17: **end while**
18: **return** $C_{max}$
19: **end procedure**

計算時間 $L_2$ だけ行う．Step1〜3 を計算時間 $CT$ を過ぎるまで繰り返し，Step2 で得られた部分グラフの最大クリークの中から最大のものを出力する．

本手法は最大クリーク探索の時間・空間計算量をそれぞれ，$O(L_2)$，$O(L_1^2)$ に緩和する．これらのパラメータは計算機環境に合わせて任意に設定できる．そのため，**3.1** で紹介した手法 [10] の時間・空間計算量 $O(2^{|V|})$，$O(|V|^2)$ に対して格段に扱いやすい．

これにより，一般的な規模（500〜2000 項目程度）のアイテムバンクから最大で 10 万個程度のテストを生成できた．

### 3.3 整数計画法を用いた最大クリークアルゴリズム

RndMCP 法はクリークを $C$ としたときに，最先端の最大クリーク探索手法 [12], [13] を用いても，$O(|C|^2)$ の空間計算量を少なくとも必要とするため，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 [14], [15]（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 $O(|C|)$ へ減少させる．ただし，この探索は $O(|C| \cdot 2^n)$ の時間計算量を必要とするため，RndMCP 法の最大クリーク探索の時間計算量 $O(L_2)$ に大幅に劣る．そこで，RndMCP 法で計算機環境のメモリの限界の頂点数 $L_1$ をも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 [14], [15],

**variables**

$$x_i = \begin{cases} 1 & i \text{ 番目の項目がテストに含まれる} \\ 0 & \text{それ以外} \end{cases}$$

**maximize**

$$\sum_{i=1}^{n} \lambda_i x_i \tag{4}$$

**subject to**

$$\sum_{i=1}^{n} x_i = \mathrm{M}(\text{ テスト項目数 }) \tag{5}$$

$$LB_{\theta_k} \le \sum_{i=1}^{n} I_i(\theta_k) x_i \le UB_{\theta_k} \tag{6}$$
$$(k = 1, ..., K)$$

$$\sum_{i=1}^{n} X_{i,r} x_i \le \mathrm{OC}(\text{ 項目重複上限数 }) \tag{7}$$
$$(r = 1, ..., |C|)$$

$$X_{i,r} = \begin{cases} 1 & i \text{ 番目の項目が} \\ & C \text{ 中の } r \text{ 番目のテストに含まれる} \\ 0 & \text{それ以外} \end{cases}$$

ここで，$\lambda_i (i = 1, 2, ..., n)$ は互いに独立な $[0, 1)$ の連続一様分布からの乱数であり，本問題が解かれるたびにリサンプリングされるものとする．

制約条件はクリーク $C$ の全頂点と隣接するための条件である．また，目的関数に含まれる $\lambda_i$ は項目 $x_i$ に対する重み付けであり，毎回ランダム組み合わせのテストが構成される．すなわち，$\lambda_i$ は項目 $x_i$ の優先順位と捉えることもできる．この定式化は Belov [21] で用いられたランダムにテスト構成を行う整数計画法への定式化を項目重複数条件について一般化したものである．

具体的には Algorithm 2 によりテスト構成を行う．

本アルゴリズムは大きく “initialize”，“add step”，“delete step” に分かれている．

“initialize” では RndMCP 法によりメモリの限界の頂点数 $L_1$ をもつグラフから最大クリーク探索し，これを初期値とする．

“add step” では前述した整数計画法で得られた解を現在探索中のクリーク $C$ へ追加することで，クリークを拡大する．これを AddCnt 回繰り返すか，整数計画法が解けなくなるまで行う．

“delete step” では現在探索中のクリーク $C$ からランダムにテストを削除することで，局所解（極大クリーク）へ収束することを回避している．計算開始からの経過時間が与えられた計算時間 CT 未満の場合は add step へ戻る．したがって，本アルゴリズムは探索中のクリークへ頂点の追加・削除を繰り返すことで，より大きなクリークを探そうとする局所探索法（local search）となっている．

本手法の時間計算量は O(CT)，空間計算量は，内部で使用する整数計画法の空間計算量が無視できるとすると，$O(|C|)$ となる．これは空間計算量を乱択法の $O(|C|^2)$ から $O(|C|)$ に減じている．したがって，乱択法と比較して，生成可能なテストの上限が大きくなる．これにより，乱択法では 10 万個のテスト構成が上限であったが，約 13 万個のテストを構成できた．

**Algorithm 2** 整数計画法を用いた最大クリーク探索

**Require:** アイテムバンク，テスト構成条件
**Ensure:** 等質テスト群
**procedure** HybridRBP($L_1, L_2, CT'$, AddCnt, $\alpha, CT$)
  $ST \coloneqq current\ time$
  /* initialize */
  $global\ C \coloneqq$ RndMCP($L_1, L_2, CT'$)
  $global\ C_{max} \coloneqq C$
  **while** $(current\ time - ST) < CT$ **do**
    /* add step */
    $count \coloneqq 0$
    **while** $count <$ AddCnt **do**
      $Sol \coloneqq$ IPSolve($C$) ▷ 式 (4)〜式 (7) を解く
      **if** $Sol \neq \emptyset$ **then** ▷ IP が解けた場合
        $C \coloneqq C \cup Sol$
        $count++$
        **if** $|C_{max}| < |C|$ **then**
          $C_{max} \coloneqq C$
        **end if**
      **else** ▷ IP が解けない場合
        break
      **end if**
    **end while**
    DeleteStep(AddCnt, $\alpha$)
  **end while**
  **return** $C_{max}$
**end procedure**
**procedure** DeleteStep(AddCnt, $\alpha$)
  /* delete step */
  $count \coloneqq 0$
  **while** $count <$ (AddCnt $\times\ \alpha$) **do**
    $C \coloneqq C \setminus \{c \in C\}$ ▷ $c$ はランダムに選択
    $count++$
  **end while**
**end procedure**

しかし，本手法は整数計画法の探索時間が大きいため，計算時間の大半を現在探索中のクリーク $C$ と隣接する頂点の探索が占めており，未だ十分な数の等質テストを生成できていない．

## 4. 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

本論文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手法を提案する．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．提案手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．

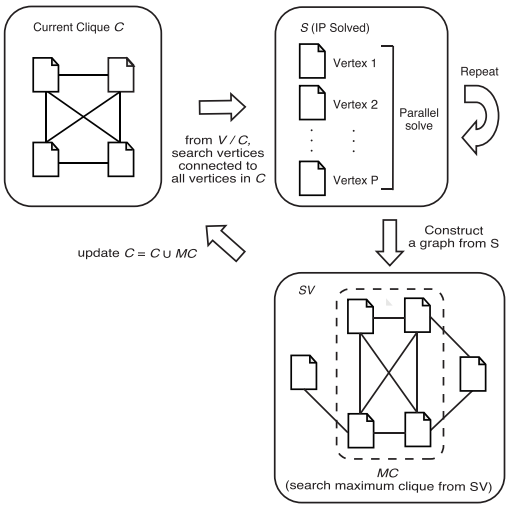


図 4　提案探索手法の概要
Fig. 4　The outline of proposed method.

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ない．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．更に，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．

図 4 は提案探索手法の概要である．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により $P$ 個探索し，候補頂点集合 $S$ の要素とする．この探索は整数計画法の制約条件を変更せずに行えるため，$P$ 個並列化できる．これを $|S|$ が $S_{\mathrm{UB}}$ となるまで繰り返す．このとき，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が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．その後，$\forall s \in S$ は探索中のクリーク $C$ 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ない．

そのため，候補頂点集合 $S$ の中から最大クリークを抽出し，探索中のクリーク $C$ に追加する．これにより，従来手法よりも多くのテストを生成できる．

具体的には Algorithm 3 により，テスト構成を行う．

本論文ではこの提案手法を PIPMCP 法（Parallel Integer Programming Maximum Clique Problem Method）と呼ぶ．

**Algorithm 3** 整数計画法を用いた最大クリーク探索の並列化

**Require:** アイテムバンク，テスト構成条件
**Ensure:** 等質テスト群
**procedure** PIPMCP($L_1, L_2, CT', S_{UB}, D_1, P, CT$)
  /* initialize */
  $global\ C \coloneqq$ RndMCP($L_1, L_2, CT'$), $C_{max} \coloneqq C$
  **while** $(current\ time - ST) < CT$ **do**
    $S \coloneqq \emptyset$
    /* search step */
    **while** $|S| < S_{UB}$ **do**
      $Sol \coloneqq \emptyset$
      **parallel for** $p \coloneqq 1 \ldots P$ **do**
        $Sol_p \coloneqq$ SearchIP($S$)
      **end parallel for**
      **if** $Sol \neq \emptyset$ **then** ▷ IP が解けた場合
        $S \coloneqq S \cup Sol$
      **else** ▷ IP が解けない場合
        DeleteStep($D_1$)
        $break$
      **end if**
    **end while**
    **if** $S \neq \emptyset$ **then**
      /* mcp step */
      $SV \coloneqq (S, E)$
      $MC \coloneqq$ MCP($SV, L_2$)
      $C \coloneqq C \cup MC$
      **if** $|C_{max}| < |C|$ **then**
        $C_{max} \coloneqq C$
      **end if**
    **end if**
  **end while**
  **return** $C_{max}$
**end procedure**
**function** SearchIP($S$)
  $lb \coloneqq 0$
  **if** $0 < |S|$ **then**
    **for** $j \coloneqq 1 \ldots |S|$ **do**
      $x \coloneqq S_j$
      $val \coloneqq \sum_{i=1}^{n} \lambda_i x_i$ ▷ 式 (4) の目的関数
      **if** $lb < val$ **then**
        $lb \coloneqq val$
      **end if**
    **end for**
  **end if**
  **return** IPSolve($C, lb$)
    ▷ 式 (4)〜式 (7) を $lb$ を下限値として解く
**end function**
**procedure** DeleteStep($D_1$)
  /* delete step */
  $count \coloneqq 0$
  **while** $count < D_1$ **do**
    $C \coloneqq C \setminus \{c \in C\}$ ▷ $c$ はランダムに選択
    $count++$
  **end while**
**end procedure**

本アルゴリズムは大きく “initialize”，“search step”，“mcp step”，“delete step” に分かれている．

“initialize” では HybridMCP 法と同様に，RndMCP 法を用いて初期値を求める．

“search step” 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 $S$ とし，要素数が $S_{\mathrm{UB}}$ となるまで整数計画法の解を追加する．HybridRBP 法では解いた整数計画法の解が次の制約条件を変更するため，並列化が困難であった．提案手法では候補頂点集合 $S$ を利用することで，整数計画法の制約条件を変更せずに行えるため，並列化ができる．更に，$0<|S|$ ならば，次の整数計画法の目的関数を最大にする $s \in S$ を求め，その値を下限値 $lb$ とする分枝限定法を用いて探索を行う．これは $\forall s \in S$ が制約条件を満たすこと，目的関数が変更されるため（$\lambda_i$ のリサンプリングが行われるため）最適値が異なることを利用している．このようにあらかじめ下限値を与えることで，分枝限定法による探索時間の削減が期待できる．

“mcp step” では候補頂点集合 $S$ の要素を頂点としたグラフから，RndMCP 法と同様に最大クリークを抽出し，探索中のクリーク $C$ へ追加する．これは，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がないために行う．

最後に，“delete step” では局所解（極大クリーク）へ収束することを回避するために，ランダムに探索中のクリーク $C$ から頂点を削除している．したがって，より大きなクリークを探そうとする局所探索法（local search）となっている．

## 5. 評価実験

提案手法（PIPMCP 法）の有効性を示すため評価実験を行う．具体的には提案手法の探索効率の評価を行った後，HybridRBP 法 [14]）とのテスト構成数を比較した．なお，実行環境は 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．

本実験には三つのシミュレーションと一つの実データによるアイテムバンクを用いた．シミュレーションは 500，1000，2000 個の項目をもち，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．また，実データは 978 項目をもつ実際に運用されていたアイテムバンクを用いた．このアイテムバンクの詳細は表 2 のとお

表 2　実アイテムバンクの詳細
Table 2　Details of actual item bank.

| Item Bank Size | Parameter $a$ | | | Parameter $b$ | | |
|---|---|---|---|---|---|---|
| | Range | Mean | SD | Range | Mean | SD |
| 978 | 0.12 ~ 3.08 | 0.43 | 0.20 | −4 ~ 4.55 | −0.22 | 1.16 |

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
Table 3　Upper and lower bound of test information for test assembly.

| $\theta = -2.0$ | $\theta = -1.0$ | $\theta = 0.0$ | $\theta = 1.0$ | $\theta = 2.0$ |
|---|---|---|---|---|
| 2.0/2.4 | 3.2/3.6 | 3.2/3.6 | 3.2/3.6 | 2.0/2.4 |

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，現在実施される規模での部分テスト構成を模倣している．ただし，項目重複数条件（OC）は OC = {0, 1, ..., 10} の 11 通りの条件によって評価する．

なお，HybridRBP 法 [14]，提案手法（PIPMCP 法）の整数計画法の探索は CPLEX [22] 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）．また，CPLEX 等のソルバーには並列分散アルゴリズムが含まれているため，計算機環境に合わせて，CPLEX で使用するスレッド数の上限を 10 とした．ただし，並列化する問題数 $P$ に合わせて，1 問に使用するスレッド数の上限を調整し，プロセッサが競合しないようにした．例えば，$P = 2$ の場合，CPLEX で使用するスレッド数の上限を 5 とし，1 問あたり最大 5 スレッドで 2 問の整数計画法を並列に解いた．

### 5.1　候補頂点集合の要素の上限数の評価

候補頂点集合の要素の上限数 $S_{\mathrm{UB}}$ の値を決定するため，$S_{\mathrm{UB}}$ がテスト構成数に与える影響を検証する．具体的には複数のテスト構成条件において，$S_{\mathrm{UB}}$ の値を変化させたとき，どのようにテスト構成数が変化するかを分析する．

PIPMCP 法のパラメータは $D_1 = 100$，$P = 1$，CT = 9hr とし，並列化を行わない．更に，$0 < |S|$ の場合において，整数計画法の下限値は与えない．これらは $S_{\mathrm{UB}}$ が与える影響のみを検証するためである．また，初期値探索の RndMCP 法で用いるパラメータは $L_1 = 100000$，$L_2 = 3$hr，$CT' = 3$hr とし，項目重複数条件が同じ条件の場合は，同じ初期値を用いた．これは，このアルゴリズムが初期値の大きさに強く影響を受けるため，初期値のテスト構成数が異なると，アルゴリズムの評価が正当に行えないためである．以降，初期値に RndMCP 法を用いる HybiridRBP 法・PIPMCP 法ではこの条件を用いる．この条件の下，候補頂点集合の上限数を $S_{\mathrm{UB}} = \{10, 100, 1000\}$ と変化させ，テスト構成数を比較した．

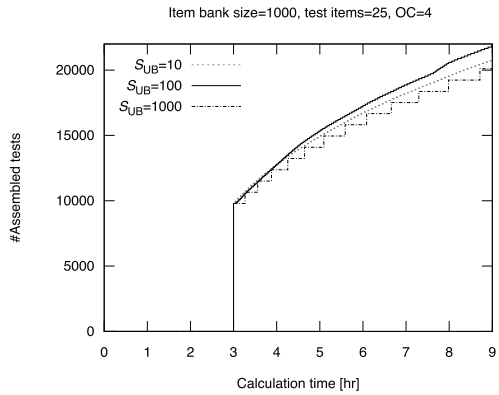


図 5　$S_{\mathrm{UB}}$ によるテスト構成数の推移
Fig. 5　The numbers of assembled tests by changing the parameter $S_{\mathrm{UB}}$.

結果を表 4 に示す．No.tests はテスト構成数，Avg. search time (MCA) [s] は $S_{\mathrm{UB}}$ の最大クリークの平均探索時間，Avg. search time (IP) [s] は整数計画法 1 回あたりの平均探索時間を示している．$S_{\mathrm{UB}} = 100$ のとき，最も多くのテストを構成できる場合が多いことがわかる．また，図 5 より，$S_{\mathrm{UB}} = 100$ とそれ以外で時間経過とともにテスト構成数の差が大きくなっている．表 4 より，OC = 6 以下のとき，$S_{\mathrm{UB}}$ の値を変化させると最大クリークと整数計画法の平均探索時間にトレードオフがある．具体的には $S_{\mathrm{UB}}$ の値を大きくすると最大クリークの探索時間は増加するが整数計画法の探索時間が減少し，$S_{\mathrm{UB}}$ の値を小さくすると逆の結果になる．ただし，OC = 7 以上ではこのトレードオフの度合いが小さくなる．以上のトレードオフが影響して，本論文の $S_{\mathrm{UB}}$ の条件においては $S_{\mathrm{UB}} = 100$ のとき，テスト構成数が多くなる場合が最も多かった．したがって，本論文では $S_{\mathrm{UB}} = 100$ を候補頂点集合の上限値として採用する．

### 5.2　整数計画法の下限値による探索効率の評価

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が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ことを示す．

表 4　$S_{UB}$ による探索効率
Table 4　The search performances of the parameter $S_{UB}$.

| Item Bank Size | OC | Proposal | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $S_{UB}$ = 10 | | | $S_{UB}$ = 100 | | | $S_{UB}$ = 1000 | | |
| | | No.tests | Avg.search time (MCA) [s] | Avg.search time (IP) [s] | No.tests | Avg.search time (MCA) [s] | Avg.search time (IP) [s] | No.tests | Avg.search time (MCA) [s] | Avg.search time (IP) [s] |
| 1000 | 0 | **34** | **0.0003** | 54.19 | **34** | 0.812 | 46.30 | **34** | 430.430 | **36.20** |
| | 1 | 264 | **0.0001** | 57.30 | **267** | 0.697 | 21.07 | 262 | 175.246 | **18.26** |
| | 2 | 1220 | **0.0002** | 20.47 | **1225** | 0.036 | 11.86 | 1171 | 180.073 | **10.20** |
| | 3 | **6345** | **0.0001** | 4.45 | 6233 | **0.001** | 3.96 | 5957 | 173.329 | **2.13** |
| | 4 | 20787 | **0.0001** | 1.88 | **21955** | **0.001** | 1.76 | 20477 | 171.324 | **1.65** |
| | 5 | 50800 | **0.0001** | 4.47 | **51048** | **0.001** | 4.28 | 51044 | 169.302 | **3.97** |
| | 6 | 91381 | **0.0001** | 10.23 | 92773 | **0.001** | 8.01 | **92825** | 0.061 | **7.69** |
| | 7 | 101349 | **0.0001** | 10.45 | **101486** | **0.001** | **10.15** | 101333 | 0.052 | 10.64 |
| | 8 | 101953 | **0.0001** | 10.68 | **101999** | **0.001** | **10.38** | 101887 | 0.051 | 10.83 |
| | 9 | 101974 | **0.0001** | 10.76 | **102028** | **0.001** | **10.50** | 101987 | 0.049 | 10.68 |
| | 10 | 101969 | **0.0001** | 10.83 | **101980** | **0.001** | **10.53** | 101923 | 0.050 | 10.87 |

表 5　下限値による探索効率
Table 5　The search performances of lower bound.

| Item Bank Size | OC | Proposal | | | | | |
|---|---|---|---|---|---|---|---|
| | | without LB | | | with LB | | |
| | | No. tests | Avg. search nodes | Avg. search time [s] | No. tests | Avg. search nodes | Avg. search time [s] |
| 1000 | 0 | **34** | 204322.9 | 46.30 | **34** | **204132.9** | **45.21** |
| | 1 | **267** | 111387.3 | 21.07 | 264 | **107970.2** | **20.94** |
| | 2 | **1225** | 52855.7 | 11.86 | 1194 | **50272.5** | **11.74** |
| | 3 | 6233 | 8378.2 | 3.96 | **6317** | **8199.2** | **3.73** |
| | 4 | 21955 | 558.2 | 1.76 | **25571** | **492.5** | **1.06** |
| | 5 | 51048 | 257.4 | 4.28 | **54146** | **203.4** | **2.43** |
| | 6 | 92773 | 195.4 | 8.01 | **94127** | **146.4** | **5.09** |
| | 7 | 101486 | 176.4 | 10.15 | **102853** | **131.7** | **5.74** |
| | 8 | 101999 | 158.2 | 10.38 | **103430** | **115.8** | **5.77** |
| | 9 | 102028 | 176.3 | 10.50 | **103449** | **126.3** | **5.33** |
| | 10 | 101980 | 192.5 | 10.53 | **103431** | **135.7** | **5.39** |

PIPMCP 法のパラメータは $D_1$ = 100，$S_{UB}$ = 100，$P$ = 1，CT = 9hr とし，並列化を行わない．この条件の下，整数計画法の下限値を与えない場合（without LB）と与えた場合（with LB）のテスト構成数を比較した．

結果を表 5 に示す．No.tests はテスト構成数，Avg. search nodes，Avg. search time [s] はそれぞれ整数計画法 1 回あたりの平均探索ノード数，平均探索時間 [s] を示している．表 5 より，OC = 3 以上になると分枝限定の効果が大きくなるため，整数計画法の探索時間が改善され，テスト構成数を増加している．図 6 は OC = 4（下限値を与えない場合と与えた場合のテスト構成数の差が最も大きい場合）のときに，横軸を計算時間，縦軸に下限値を与えない場合と与えた場合のテスト構成数を示している．図 6 より，時間経過とともに下限値による分枝限定の効果が大きくなっていることがわかる．しかし，表 5 における OC = 2 以下の場合では下限値を与えてもテスト構成数に改善がみられない．これは分枝限定の効果が小さく，整数計画法の探索速度が改善されていないことを示している．前述したように，提案手法は整数計画法の目的関数におけ

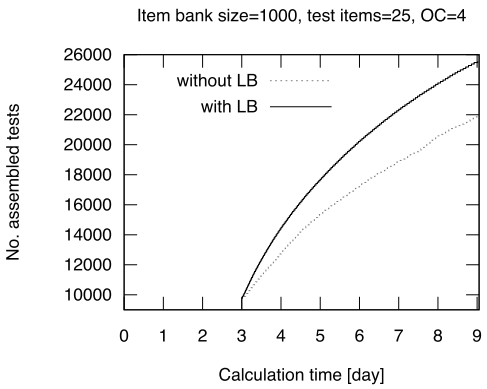


図 6　下限値によるテスト構成数の比較
Fig. 6　The number of assembled tests of lower bound.

表 6　並列探索による効率
Table 6　The performances of parallel solve.

| Item Bank Size | OC | Proposal | | | |
|---|---|---|---|---|---|
| | | $P = 1$ | $P = 2$ | $P = 5$ | $P = 10$ |
| 1000 | 0 | **34** | **34** | **34** | **34** |
| | 1 | 264 | 265 | **267** | 261 |
| | 2 | 1194 | 1194 | **1205** | 1194 |
| | 3 | **6317** | 6288 | 6179 | 6058 |
| | 4 | 25571 | 28714 | **30350** | 28517 |
| | 5 | 54146 | 58908 | 64616 | **66704** |
| | 6 | 94127 | 96825 | 100323 | **102125** |
| | 7 | 102853 | 105296 | 108356 | **110046** |
| | 8 | 103430 | 105879 | 108829 | **110579** |
| | 9 | 103449 | 105898 | 108888 | **110498** |
| | 10 | 103431 | 105884 | 108785 | **110500** |

る $\lambda_i$ に乱数を用いており，テスト構成数にばらつきが生じる．このばらつきにより，分枝限定の効果が小さいときには下限値を与えていない方が与えるよりもテスト構成数が多くなってしまう場合がある．この乱数による影響を緩和するために，分枝限定の効果が小さい OC = $\{0,1,2\}$ の条件について，同様の実験を 5 回行い，テスト構成数の平均値を算出した．結果は下限値を用いない場合の OC = $\{0,1,2\}$ でのテスト構成数の平均値が $\{33.8, 264.4, 1216.0\}$ であったのに対して，下限値を用いた場合が $\{34.0, 264.8, 1217.2\}$ であった．t 検定では有意差が見られなかったが，下限値を用いた場合のテスト構成数の平均値が 0.2〜1.2 高くなり，わずかに良い値を示した．以降，提案手法は下限値を用いてテスト構成を行う．

### 5.3　並列化による探索効率の評価

並列化による探索効率を評価する．なお，PIPMCP 法のパラメータは $D_1 = 100$，$S_{\mathrm{UB}} = 100$ とし，並列化数を $P = \{1,2,5,10\}$ と変化させ，テスト構成数を比較した．

結果を表 6 に示す．OC=3 以下の場合にテスト構成数の変化が小さいのは，整数計画法の探索時間と並列化数 $P$ のトレードオフが大きく，並列化しても得られる解の総数が変わらないためと考えられる．しかし，OC = 4 以上ではこのトレードオフが小さくなり，並列化数 $P$ を増やした方が得られる解の総数が多くなると考えられる．これには **5.2** で示したように，下限値を与える効果が大きくなることも関係していると考えられる．

したがって，本手法は計算機を複数台用いた並列処理も可能であるが，できるだけ多くの問題を並列して解いた方が最終的に得られる解の総数が多くなることを示している．

ただし，テストの構成条件に合わせて，最適な並列化数 $P$ を決定することは容易ではない．例えば，Koch et al. [23] によれば，整数計画法で用いるスレッド数に応じて，一概には探索する分枝数や計算時間が改善されないことを実験的に示している．したがって，このトレードオフは OC の条件に依存していると考えられるため，適切な値が定まらない．そのため，OC = 4 以上でテスト構成数の増加が顕著であった $P = \{5,10\}$ を用いて，次節では従来手法との比較を行う．

### 5.4　従来手法との比較

提案手法（PIPMCP 法）の有効性を示すため，従来手法（HybridRBP 法 [14]）とテスト構成数を比較した．

各手法の計算時間は 24hr とし，HybridRBP 法には AddCnt = 1000，$\alpha = 10\%$，CT = 24hr，PIPMCP 法には $D_1 = 100$，$S_{\mathrm{UB}} = 100$，CT = 24hr，並列化数 $P$ は $P = \{5,10\}$ と 2 種類の条件で行った．

結果を表 7 に示す．提案手法は OC の値が大きくなると，徐々に有効性が現れる．例えば，アイテムバンクサイズが 1000 の場合，OC が 4 以上になると提案手法が最も多くのテストを構成する．ただし，OC が 3 以下になると，HybridRBP 法が最も多くのテストを構成している．これは，提案手法が候補頂点集合中の最大クリークを追加する必要があるため，クリーク $C$ に追加しない頂点も求めているためである．特に，構成テスト数が少ない条件（OC が小さい）では候補頂点集合中の最大クリークの大きさが小さいため，クリーク $C$ に追加しない頂点の割合が大きい．例えば，アイテムバンクサイズ 500，項目重複数条件 OC=3 の条件において，提案手法が計算途中に整数計画法で求めた解の総数を調べると，HybridRBP 法の 921 に対して，1305（$P=5$），1230（$P=10$）と上回っていたが，その内，クリーク $C$ に追加された解の総数は 598（$P=5$），568（$P=10$）と 50% 未満であった．したがって，このような条件においては HybridRBP 法のように一つずつ解を求めてクリーク $C$ に追加した方が，テスト構成数が多くなる．

提案手法はテスト構成数が大きい場合に，有効性が顕著に現れる．そこで，（アイテムバンクサイズ，重複項目数条件）=（2000,5）の条件において，計算時間を 672hr（28 日間）まで延長したときの PIPMCP 法と Hybirid 法のテスト構成数を比較した．

図 7 は構成テスト数の推移をプロットしたものである．図中の破線が HybridRBP 法，実線が提案手法を表している．図 7 のとおり，提案手法は時間経過とと

表 7　大規模アイテムバンクにおけるテスト構成数の比較
Table 7　The numbers of assembled tests for each methods in large scale item banks.

| Item Bank Size | OC | Hybrid RBP | Proposal | |
|---|---|---|---|---|
| | | | $P = 5$ | $P = 10$ |
| 500 | 0 | **17** | **17** | **17** |
| | 1 | **47** | 42 | 41 |
| | 2 | **267** | 237 | 240 |
| | 3 | **1144** | 773 | 730 |
| | 4 | **5032** | 3471 | 3348 |
| | 5 | 12550 | **14874** | 14331 |
| | 6 | 29207 | **54955** | 49837 |
| | 7 | 67969 | **98026** | 97792 |
| | 8 | 98406 | 121916 | **122378** |
| | 9 | 104991 | 126649 | **127229** |
| | 10 | 105002 | 126987 | **127149** |
| 1000 | 0 | **34** | **34** | **34** |
| | 1 | **318** | 266 | 264 |
| | 2 | **1892** | 1284 | 1240 |
| | 3 | **7557** | 6891 | 6771 |
| | 4 | 20653 | **37831** | 35731 |
| | 5 | 55024 | 93212 | **97492** |
| | 6 | 96527 | 119923 | **125324** |
| | 7 | 106834 | 120046 | **131966** |
| | 8 | 107942 | 127159 | **131579** |
| | 9 | 107735 | 127253 | **131948** |
| | 10 | 107672 | 127432 | **131300** |
| 2000 | 0 | **70** | **70** | **70** |
| | 1 | **1531** | 1035 | 988 |
| | 2 | 6963 | **7662** | 7569 |
| | 3 | 25364 | **54168** | 51401 |
| | 4 | 72520 | 101780 | **108165** |
| | 5 | 103354 | 124055 | **129257** |
| | 6 | 106362 | 125991 | **131791** |
| | 7 | 107434 | 126863 | **132273** |
| | 8 | 107774 | 126685 | **132090** |
| | 9 | 107998 | 126245 | **133550** |
| | 10 | 107783 | 126532 | **140700** |
| 978 | 0 | **35** | **35** | **35** |
| | 1 | **348** | 298 | 286 |
| | 2 | **1844** | 1340 | 1334 |
| | 3 | 6960 | **7236** | 7050 |
| | 4 | 14866 | **34031** | 31724 |
| | 5 | 52126 | **80430** | 73693 |
| | 6 | 93704 | **113039** | 108935 |
| | 7 | 104339 | **121503** | 118165 |
| | 8 | 105823 | **122167** | 119797 |
| | 9 | 105805 | **122258** | 119758 |
| | 10 | 105956 | 122681 | **124200** |

もに，HybridRBP 法とテスト構成数の差が広がる．また，672hr（28 日間）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．最終的に HybridRBP 法は生成されるテスト数が 194575 個であったのに対し，提案手法では 438950 個と，テスト生成数が約 2.2 倍にテスト生成数を更新した．これは年間 20 万人以上が

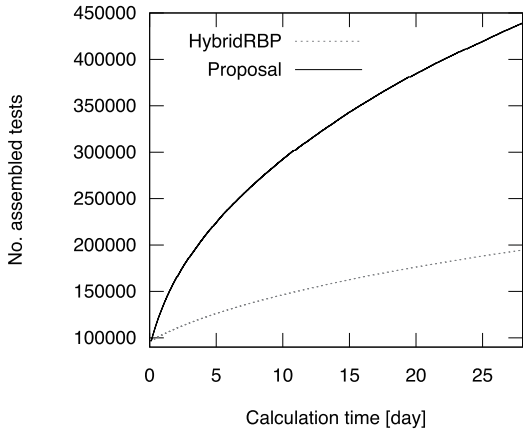


図 7　計算時間 672hr（28 日間）の提案手法と HybridRBP 法のテスト構成数
Fig. 7　The number of assembled tests by HybridRBP and Proposal in 672hr (28 days).

受験している情報処理技術者試験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 HybridRBP 法が 4 週間かかるテスト生成を 1 週間足らずで生成でき，計算時間を改善できた．ただし，提案手法は **5.1**「候補頂点集合の要素の上限数の評価」で示した $S_{\mathrm{UB}}$ の値による最大クリークと整数計画法の平均探索時間及び **5.3**「並列化による探索効率の評価」で示した整数計画法の探索時間と並列化数 $P$ についてのトレードオフの問題に帰着する．これらの条件は実験により最適な値を求める必要があり，計算機環境に依存することに留意してほしい．

## 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構成のための最大クリーク探索を行う並列アルゴリズムを提案した．本手法は候補頂点集合を導入し，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ことで探索効率を改善した．更に，候補頂点集合の要素を次に解く整数計画法の下限値とすることで探索をより高速化した．これにより，少なくともテスト間の重複項目数が 5 以上の条件においては提案手法に有効性があることをシミュレーションデータ・実データを用いて示した．ただし，430000 個程度のテスト構成に 4 週間を要するため，今後もより効率的な手法を開発する必要がある．

また，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．実際に，438950 個のテスト

群ではある項目が約 10000 個のテストに含まれている一方で，約 200 個のテストにしか含まれていない項目もある．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 [16]．そのため，この露出を制御できる手法を検討する．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 [24]．

更に，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 [25], [26]．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**　本研究の一部は科学研究費補助金（基盤研究（S），代表：植野真臣）「信頼性向上を持続する e テスティング・プラットフォームの開発」（19H05663）の助成を受けた．

## 文　献

[1] 植野真臣，永岡慶三，e テスティング，培風館，2009.
[2] 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.
[3] 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.
[4] P. Songmuang and M. Ueno, “Bees algorithm for construction of multiple test forms in e-testing,” IEEE Trans. Learning Technologies, vol.4, no.3, pp.209–221, 2010.
[5] W.J. van der Linden, Liner Models for Optimal Test Design, Springer, 2005.
[6] E. Boekkooi-Timminga, “The construction of parallel tests from irt-based item banks,” J. Educational Statistics, vol.15, no.2, pp.129–145, 1990.
[7] R.D. Armstrong, D.H. Jones, and Z. Wang, “Automated parallel test construction using classical test theory,” J. Educational Statistics, vol.19, no.1, pp.73–90, 1994.
[8] R.D. Armstrong, D.H. Jones, and C.S. Kunce, “Irt test assembly using network-flow programming,” Applied Psychological Measurement, vol.22, no.3, pp.237–247, 1998.
[9] W.J. van der Linden and J.J. Adema, “Simultaneous assembly of multiple test forms,” J. Educational Measurement, vol.35, no.3, pp.185–198, 1998.
[10] 石井隆稔，植野真臣他，“最大クリーク問題を用いた複数等質テスト自動構成手法，”信学論（D），vol.97, no.2, pp.270–280, Feb. 2014.
[11] T. Ishii, P. Songmuang, and M. Ueno, “Maximum clique algorithm and its approximation for uniform test form assembly,” IEEE Trans. Learning Technologies, vol.7, no.1, pp.83–95, 2014.
[12] E. Tomita, S. Matsuzaki, A. Nagao, H. Ito, and M. Wakatsuki, “A much faster algorithm for finding a maximum clique with computational experiments,” J. Information Processing, vol.25, pp.667–677, 2017.
[13] C.M. Li, H. Jiang, and F. 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] 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，”信学論（D），vol.100, no.1, pp.47–59, Jan. 2017.
[15] T. Ishii and M. Ueno, “Algorithm for uniform test assembly using a maximum clique problem and integer programming,” Int. Conf. Artificial Intelligence in EducationSpringer, pp.102–112, 2017.
[16] H. Wainer, “Cats: Whither and whence,” Psicologica, vol.21, no.1, pp.121–133, 2000.
[17] D.I. Belov and R.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.
[18] F.M. Lord and M.R. Novick, Statistical theories of mental test scores, IAP, 2008.
[19] F.B. Baker and S.H. Kim, Item response theory: Parameter estimation techniques, CRC Press, 2004.
[20] K.T. Sun, Y.J. Chen, S.Y. Tsai, and C.F. Cheng, “Creating irt-based parallel test forms using the genetic algorithm method,” Applied measurement in education, vol.21, no.2, pp.141–161, 2008.
[21] D.I. Belov, “Uniform test assembly,” Psychometrika, vol.73, no.1, p.21, 2008.
[22] IBM, “Ilog cplex optimization studio cplex user’s manual 12.9,” 2019.
[23] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R.E. Bixby, E. Danna, G. Gamrath, A.M. Gleixner, S. Heinz, et al., “Miplib 2010,” Mathematical Programming Computation, vol.3, no.2, p.103, 2011.
[24] T. Ishii and M. Ueno, “Clique algorithm to minimize item exposure for uniform test forms assembly,” International Conference on Artificial Intelligence in EducationSpringer, pp.638–641, 2015.
[25] 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，”信学論（D），vol.101, no.6, pp.909–920, June 2018.
[26] M. Ueno and Y. Miyazawa, “Uniform adaptive testing using maximum clique algorithm,” International Conference on Artificial Intelligence in EducationSpringer, pp.482–493, 2019.

（2020 年 1 月 17 日受付，6 月 1 日再受付，8 月 5 日早期公開）

**渕本　壱真**

2020 年電気通信大学情報理工学域・先端工学基礎課程卒．同年，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入学，現在に至る．e テスティング，等質テストの研究・開発に従事．

**植野　真臣**　（正員）

1992 神戸大学大学院教育学研究科修了，1994 東京工業大学大学院総合理工学研究科了．博士（工学），東京工業大学，千葉大学，長岡技術科学大学を経て 2006 より電気通信大学勤務，2013 教授．2016 電気通信大学大学院情報理工学研究科教授，現在に至る．電気通信大学大学院情報システム学研究科教授．人工知能，e テスティング，e ラーニング，ベイズ統計の研究に従事．2008 The 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008) Best Paper Award．2017 電子情報通信学会論文賞．