BIB デザイン生成のための推論システム

慶應大・理工　佐藤 秀
慶應大・理工　大原 幸多
慶應大・理工　神保 雅一

1　はじめに

従来, 実験計画とともに発展してきた組合せデザインは, 符号理論 · 暗号などとも関係が深く, 最近では DNA 解析などにも応用研究がなされている. 本論文では, このような応用例で使用される, パラメータの大きなデザインを探索するための推論システムについて報告する.

1.1　組合せデザイン探索のための推論システムの意義

組合せデザインの 1 つである 'balanced incomplete block (BIB) デザイン' が農業や工業の実験計画の分野で用いられることはよく知られている. また, BIB デザインと符号理論, 暗号およびコンピュータネットワークなどとの関係についても多くの研究者によって研究されてきた. 最近では, 鍵共有暗号システム, 光 CDMA (code division multi access) のための光直交符号, DNA 解析のための grid デザインなどの新しい組合せデザインの応用についても研究されている. しかし, このような応用例では, パラメータの大きなデザインを用いることが多く, Clatoworthy(1973) や統計学事典 (1989) などにまとめられている BIB デザインのパラメータの表では対応することができない.

このため, すでに知られている構成定理や存在 · 非存在定理などをデータベースとして貯蓄し, それらをもとに推論を行い, ユーザにより要求されたパラメータを持つ BIB デザインを探索, 構成する推論システム 'Design Navigation system (DesNavi)' を開発した.

さらに, BIB デザインの存在·非存在の探索のみならず, 種々の組合せデザインにも対応できるように, 以下の 5 点を追加した.

(i) BIB デザインについては, 射影幾何, アフィン幾何, アダマールデザイン, cyclic difference family, cyclic difference set による実際のブロックの構成法.
(ii) packing デザインについては, 最大ブロック数のとりうる値の計算法.
(iii) covering デザインについては, 最小ブロック数のとりうる値の計算法.
(iv) group divisible デザインの探索.
(v) group divisible デザインを用いた BIB デザインの構成定理.

DesNavi を BIB デザインだけでなく, より一般の組合せデザインに対応させることが可能となった.

1.2 種々の組合せデザインを適用した DesNavi の構築

上述の推論システムの構築を行うために, 本論文で取り扱う種々のデザインについて定義や定理を述べる.

定義 1 V を v 個の要素からなる集合, $\mathcal{B}$ を V 上の k 個の要素からなる部分集合の族とする. V および $\mathcal{B}$ の要素をそれぞれ点, ブロックと呼ぶ. V 上の異なる 2 つの点の組が $\mathcal{B}$ に属するちょうど λ 個のブロックに含まれるとき, 組 $(V, \mathcal{B})$ を **balanced incomplete block (BIB) デザイン**もしくは, **2-デザイン**と呼び, $B[k, \lambda; v]$ と表す.

このとき, λ を会合数といい, λ=1 の BIB デザインを **Steiner 2-デザイン**と呼ぶ. BIB デザインでは，ある 1 点を含むブロックの数は, 点の選び方によらず一定である (そのブロック数を繰り返し数 r とおく). このとき,

$$vr = bk \text{ かつ } \lambda(v-1) = r(k-1)$$

が成り立つことは容易にわかる. ここで, b は $\mathcal{B}$ に属するブロックの総数である.

定理 1 *BIB* デザイン $B[k, \lambda; v]$ が存在するための必要条件は

$$\lambda(v-1) \equiv 0 \pmod{k-1} \text{ かつ } \lambda v(v-1) \equiv 0 \pmod{k(k-1)}. \quad (1.1)$$

一般の場合, つまり, パラメータ v, k, λ をもつデザインが条件式 (1.1) を満たさない場合については, BIB デザインは存在しない. そのような場合を扱うために, packing デザインと covering デザインを紹介する.

定義 2 V, $\mathcal{B}$ をそれぞれ v 個の点集合, V の k 個の点からなる部分集合の族とする. V 上の異なる 2 点を含む $\mathcal{B}$ 上のブロックが最高 λ 個存在するとき, 組 $(V, \mathcal{B})$ を **packing デザイン** と呼び, $PD[k, \lambda; v]$ と表す. 逆に, V 上の異なる 2 点を含む $\mathcal{B}$ 上のブロックが最低 λ 個存在するとき, 組 $(V, \mathcal{B})$ を **covering デザイン** と呼び, $CD[k, \lambda; v]$ と表す.

ここで, packing (covering) デザインについては, ブロックの総数 b, 繰り返し数 r は一定にはならないが, ブロックの総数が多い (少ない) 方が好ましく, 多くの数学者によって $PD[k, \lambda; v]$, $CD[k, \lambda; v]$ のブロック数のそれぞれ最大数, 最小数が研究されている.

定理 2 (Johnson Bound) $\sigma(k, \lambda; v)$ を $PD[k, \lambda; v]$ の最大ブロック数とする. このとき,

$$\sigma(k, \lambda; v) \leq U(k, \lambda; v) \quad (1.2)$$

が成り立つ. ここで,

$$U(k, \lambda; v) = \left\lfloor \frac{v}{k} \left\lfloor \frac{v-1}{k-1} \lambda \right\rfloor \right\rfloor \quad (1.3)$$

であり, $\lfloor x \rfloor$ は $\lfloor x \rfloor \leq x$ を満たす最大の整数である.

定理 3 (Schönheim Bound) $\alpha(k,\lambda;v)$ を $CD[k,\lambda;v]$ の最小ブロック数とする. このとき,

$$\alpha(k,\lambda;v) \geq L(k,\lambda;v) \tag{1.4}$$

が成り立つ. ここで,

$$L(k,\lambda;v) = \left\lceil \frac{v}{k} \left\lceil \frac{v-1}{k-1}\lambda \right\rceil \right\rceil \tag{1.5}$$

であり, $\lceil x \rceil$ は $\lceil x \rceil \geq x$ を満たす最小の整数である.

(1.2), (1.4) 式で, それぞれ

$$\sigma(k,\lambda;v) = U(k,\lambda;v),\ \ \alpha(k,\lambda;v) = L(k,\lambda;v)$$

となる packing デザイン, covering デザインを **optimal** であるという.

さらに, 次に紹介する 'group divisible デザイン' や orthogonal array は, BIB デザインを構成するのにもよく用いられている.

定義 3 $V, \mathcal{B}$ をそれぞれ v 個の点集合, V の k 個の点からなる部分集合の族とし, $\mathcal{G}$ を V の分割とする. つまり, $\mathcal{G}$=$\{G_1, G_2, \cdots, G_n\}$, $|G_i|$=g $(1 \leq \forall i \leq n)$, $G_i \cap G_j$=ϕ $(i \neq j)$, $\cup_{i=1}^{n} G_i$=V. $\mathcal{G}$ の要素をグループといい, g をグループサイズ, n をグループ数という. V 上の異なる 2 点が, $\mathcal{G}$ 上の同じグループに属するとき, その 2 点を含む $\mathcal{B}$ 上のブロックは存在しない, また $\mathcal{G}$ 上の異なるグループに属するとき, その 2 点を含む $\mathcal{B}$ 上のブロックはちょうど λ 個存在するとき, 組 $(V,\mathcal{B})$ を **group divisible (GD) デザイン** と呼び, $GDD[k,\lambda,g;v]$ と表す (v=ng である). また, n=k の GD デザインを **transversal デザイン** といい, $TD[k,\lambda,g;v]$ と表す (v=kg である).

定義 4 s 種類の文字を要素とする λs^t 行 m 列の行列を A とする. A の任意の λs^t 行 t 列の部分行列について, s 種類の文字を要素とする s^t 通りのすべての 1 行 t 列がちょうど λ 回ずつ現れるとき, この行列 A を **直交配列**(**orthogonal array, OA**) と呼び, $OA(t,m,s,\lambda)$ と表す.

$t = 2$ のとき, $OA(2, s, m, \lambda)$ の任意の 1 行 t 列 $(x_1, x_2, \cdots, x_m)$ の i 番目の成分 x_i を i 番目のグループの x_i 番目の要素に対応させると, $\{x_1, x_2, \cdots, x_m\}$ をブロックとしてもつ transversal デザインになる. つまり,

定理 4 直交配列 $OA(2, s, m, \lambda)$ が存在する. $\Longleftrightarrow$ $TD[m, \lambda, s; ms]$ が存在する.

以上の性質をもとに, 関連する種々の定理を推論システムに組み込み構築を行った.

2 DesNavi: Design Navigation system

本研究において構築した組合せデザイン探索のための推論システムは, 'Design Navigation system' 略して, 'DesNavi' と名づけられている. 本章では, このシステム 'DesNavi' の構造について述べる.

2.1 DesNaviの機能

DesNavi では, 次のことが実行できる.

(i) BIB デザインの探索, および構成

BIB デザインは竹内ら (1989) や, 多くの研究者に研究され, Beth, Jungnichel and Lenz (1986), および Mathon and Rosa(1996, 改訂 2000) によりそのパラメータは表にまとめられている. これらの表にあげられたパラメータについてはもちろん, 載っていないパラメータについても, 存在·非存在の定理群をチェックし, '存在', '非存在', '未知' のいずれかを返す (このとき, 構成法も出力する).

また, BIB デザインが存在するときには, 複数の構成法を挙げること

ができる (非同型なデザインであるかはまだ判別していない).

さらに, 一部だが BIB デザインの実際のブロックを生成することができる. 現在, 射影幾何, アフィン幾何, アダマールデザインによる構成法からなる BIB デザインや cyclic difference family, cyclic difference set による構成法からなる BIB デザインについてのみブロックを生成することができる.

(ii) packing/covering デザインの最大/最小ブロック数の計算

packing/covering デザインについては, Colbourn and Dinitz (1996) に載っている定理はすべて組み込んであり, それぞれ最大/最小ブロック数のとりうる値の範囲を返す. また, optimal であるかどうかも返す.

(iii) group divisible デザインの探索

group divisible デザインについては, '存在', '非存在', '未知' のいずれかを返す. また, k, λ, v の値を変えずに, グループサイズ g, グループ数 n が異なる group divisible デザインが存在するときにはそれも表示することができる.

DesNavi には, 種々のデザインについて, のべ 130 の定理が組み込まれている. このうち, BIB デザインに関して 10, packing デザインに関して 25, covering デザインに関して 20, group divisible デザインに関して 30, transversal デザインに関しては 15 ほどの定理は本研究で新しく追加した定理であるである. また, 入力パラメータの値が 10000 を超過しても, 結果を迅速に返す.

2.2 DesNavi の構造

DesNavi は (1) サーバとクライアント間の通信部分にはネットワークに適した Java プログラミング言語, (2) 有限体の演算, 因数分解等の計算用として Mathematica, さらに, (3) 核の部分は Java プログラミング言

語, Mathematica との通信が可能で, かつ処理速度の速い C プログラミング言語を用いることによって実現している. Mathematica の中の関数は Mathlink library の使用により C プログラミング言語から呼び出すことができる.

2.3 DesNavi のクライアント

本システムはサーバとクライアント間の通信に Java アプレットを用いている. Java アプレットはサーバからの Web ブラウザにダウンロードされる Java クラスであり, ブラウザの管理下で実行される. 一方, Java アプリケーションは単独で実行することができる Java プログラムです. Java アプレットを使用しているのはクラスファイルを事前にダウンロードしていなくても扱うことができるからである. つまり, ユーザは Web ブラウザだけあれば, Java 環境を準備しなくてもよいのである. ユーザが DesNavi の Web ページを開くと, クライアントアプレットはユーザのブラウザにダウンロードされる. そこで, ユーザが調べたいデザインのパラメータを入力する.

2.4 DesNavi のサーバ

DesNavi サーバにおいては, 3 つのプログラムが常に実行されている. 1 つ目は 'DesignServer.class' と呼ばれる Java アプリケーションプログラムである. 2 つ目は 'Design.c' と呼ばれる C 言語プログラムである. また, 3 つ目は 'MathematicaKernel' である. Java アプリケーションはクライアント上の Java アプレットからパラメータを受け取り, それを C プログラミング言語によって書かれたメインルーチン (Design.c) にパラメータを転送する.

2.5 サーバ上の定理

本システムでは, すべての定理はC言語の関数として扱っている. まず非存在定理をチェックし, 定理に当てはまる場合, つまりデザインが存在しない場合は'非存在'を示す値 (-1) を返す. 次に存在定理をチェックする. 存在定理の中には, 再帰的にチェックをするものもある. ここでデザインが存在する場合は, '存在'を示す値 (1) を返す. 以上の定理に当てはまらないときは, '未知'を示す値 (0) を返す.

また, より効率よく探索するために, 探索する定理の優先順位を, その使用頻度により動的に組替えている.

ここで, 関数の例をひとつ挙げる.

例 1 逐次構成法はDesNaviではC言語の関数'et3()' (existence theoremの意) に対応している. 逐次構成法とは, $B[k, \lambda_1, v_1]$, $B[k, \lambda_1\lambda_2; v_2]$ および $OA(2, k, v_2, \lambda_2)$ が存在するならば, $B[k, \lambda_1\lambda_2; v_1v_2]$ は存在する, という定理である. まず v の因数を求めるため, 以下のプログラムにある3行目の v を素因数分解する関数 `getfactor(v)` を用い, 結果を配列 `factor[][]` に格納する. その後, 考えうるすべての因数 v_1, v_2 の組み合わせについて, ひとつひとつ $B[k, \lambda_1, v_1]$, $B[k, \lambda_1\lambda_2; v_2]$ が存在するか否かをチェックする.

--------- BIB デザインの逐次構成法に関するプログラム---------

```
 1:int et3(int v, int k, int lambda){
 2:   int v1,v2,lambda1,lambda2;
 3:   int i,j,p,cnt=0;
 4:   int factorlist[40], primelist[40];
 5:   getfactor(v);
 6:   for(j=1;j<=answerfactor[0][0];j++) {
 7:      for(i=1;i<=answerfactor[j][1];i++){
 8:         primelist[cnt]=answerfactor[j][0];
```

```
 9:         factorlist[cnt]=powerqton(answerfactor[j][0],i);
10:         cnt ++;
11:      }
12:   }
13:   for(p=0;p<cnt;p++){
14:      v2 = factorlist[p];
15:      if(k<=(lambda*v2*v2-1)/(v2-1)){
16:         if(primepowercheck(v2)==1){
17:            for(lambda1=1;lambda1<=l;lambda1++){
18:               if(v%v2==0 && lambda%lambda1==0){
19:                  v1=v/v2;
20:                  if(v1==1 || v1<k || v2+1<k || primelist[p]< k){
21:                  }else{
22:                     if(design(v1, k, lambda1)==1){
23:                        printf("First BIBD found !!\n");
24:                        lambda2=lambda/lambda1;
25:                        if(design(v2, k, lambda2)==1){
26:                           printf("Second BIBD found !!\n");
27:                           return 1;
28:                        }
29:                     }
30:                  }
31:               }
32:            }
33:         }
34:      }
35:   }
36:   return 0;
```

```
37:}
```

2.6 Mathmatica とのリンク

DesNaviでは, 因数分解や有限体の演算等の計算用として, Mathematica を用いている. パラメータの計算, およびデザインの構成には Fujiwara (1998) の作った Galois Field package を組み込んでいる. 先の例 1 でも紹介したC言語の関数 getfactor(v) は Mathlink library を通じて Mathematica を呼び出して実行する.

-----Mathematica を呼び出し, 因数分解をするプログラム-----

```
 1:int getfactor(int n){
 2:   int pkt, prime, expt; long len;
 3:   MLPutFunction(lp, "EvaluatePacket", 1L);
 4:   MLPutFunction(lp, "FactorInteger", 1L);
 5:   MLPutInteger(lp, n); MLEndPacket(lp);
 6:   if (!MLCheckFunction(lp, "List", &len)) error(lp);
 7:   factor[0][0]=len;
 8:   for (k=1; k<len+1; k++) {
 9:      if (MLCheckFunction(lp, "List", &lenp) && lenp==2
10:        && MLGetInteger(lp, &prime) &&  MLGetInteger(lp, &expt)){
11:         factor[k][0] = prime;
12:         factor[k][1] = expt;
13:      }
14:   }
15:   return factor[0][0];
16:}
```

4 行目の MLPutFunction(FactorInteger) で FactorInteger のコマンドを Mathematica から呼び出し, 5 行目の MLPutInteger で引数 n の値を代入する. 素因数分解の結果, k 番目の素数およびその指数をそれぞれ factor[k][0], factor[k][1] に格納する (11, 12 行目). また, 素因数分解したときの素数の個数を factor[0][0] に格納する (7 行目).

例えば, FactorInteger[5913] を実行すると,

$$\begin{aligned} &\text{factor}[0][0] = 2 \\ &\text{factor}[1][0] = 3, \quad \text{factor}[1][1] = 4; \\ &\text{factor}[2][0] = 73, \quad \text{factor}[2][1] = 1; \end{aligned}$$

となる. これは $5913 = 3^4 * 73^1$ を意味する.

3 DesNavi の使用法

本章では, ネット上に公開されている DesNavi の使用法を説明する. また, 具体例を通して, その出力結果の見方を説明する.

3.1 DesNavi の使用法

DesNavi はネット公開されており, 慶応大学神保研究室のホームページ (http://jim.math.keio.ac.jp) からリンクが張ってある. 以下の 7 つのステップに沿って, DesNavi を操作する.

Step 1. 慶応大学神保研究室のホームページより, DesNavi に入る. 図 1 は DesNavi のクライアント画面である.

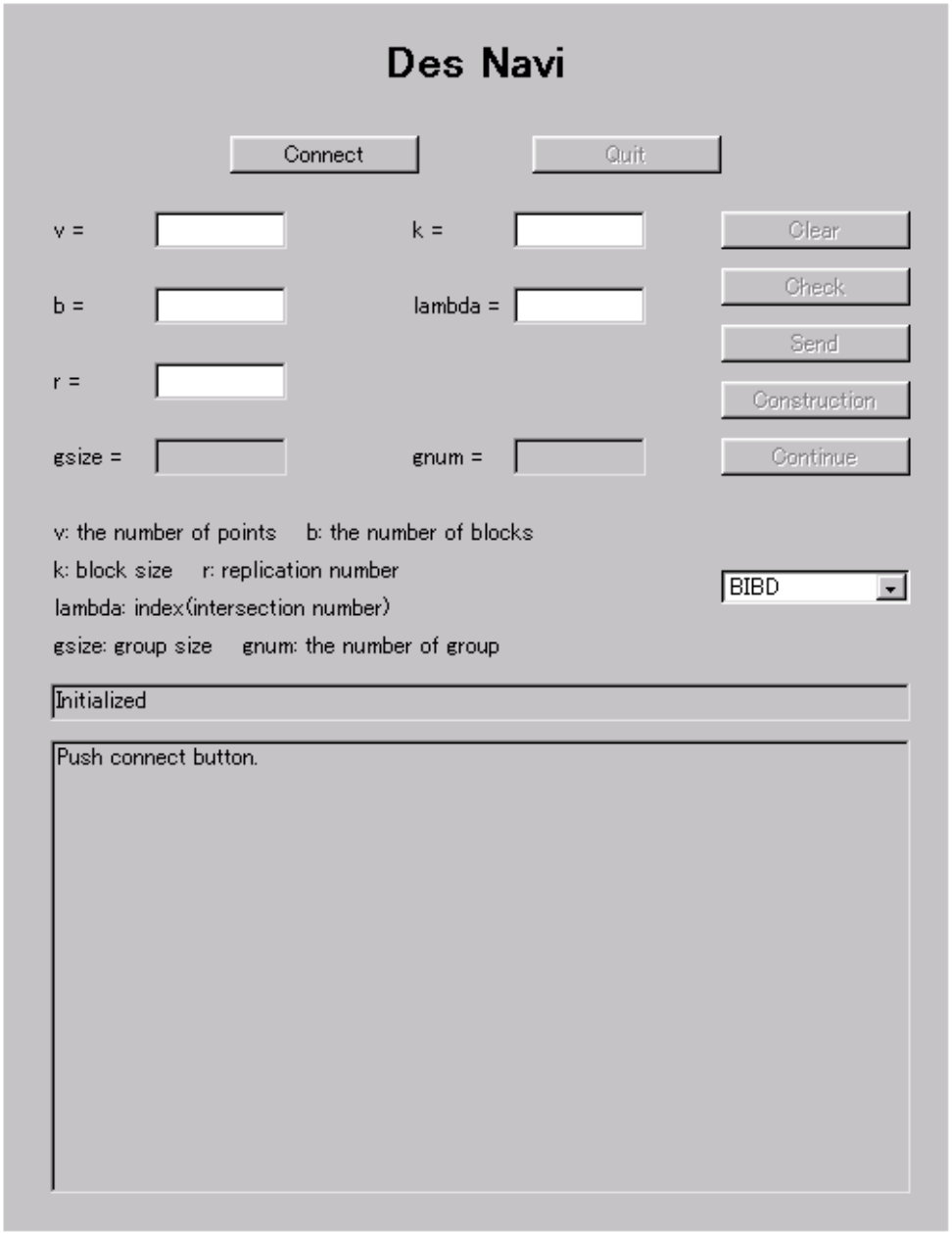

図 1: Step 1　　DesNavi のクライアント画面

Step 2. まず, 'connect' ボタンを押し, サーバと接続する. 接続が完了したら, 図 2 のように表示が出る. ここで, '1 persons connected.' とはサーバと接続している人数を意味する. つまり, この場合, ユーザ本人だ

けがサーバと接続している.

図 2: Step 2　　サーバとの接続

Step 3. 調べたいデザインのパラメータを入力する. ここで, 調べたいデザインが BIB デザインのときは, パラメータ v, k, λ, b, r のうち少なくとも 3 つ入力する. packing/covering デザインのときは, v, k, λ を入力する. group divisible デザインのときは, v, k, λ を入力, もしくは, v, グループサイズ g, グループ数 n の少なくとも 2 つと k, λ, を入力する. パラメータを入力後, 'check' ボタンを押す.

パラメータ v=25, k=5, λ=1 をもつ BIB デザインを探索する場合を例として挙げる. 図 3 はパラメータ v=25, k=5, λ=1 を代入したところである.

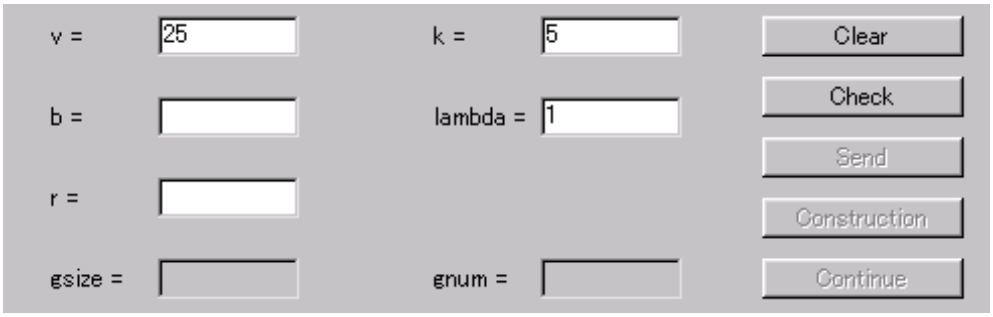

図 3: Step 3　　パラメータの代入

Step 4. 入力したパラメータが, デザインが存在するための必要条件を満たしているかどうかを 'check' ボタンを押してチェックする. このチェックはクライアントアプレットが行っている. BIB デザインが存在するための必要条件 (1.1) が成り立つときは 'OK' が表示され, 入力されなかったパラメータを計算しその値も表示する. 図 4 では b, r の値が計算された.

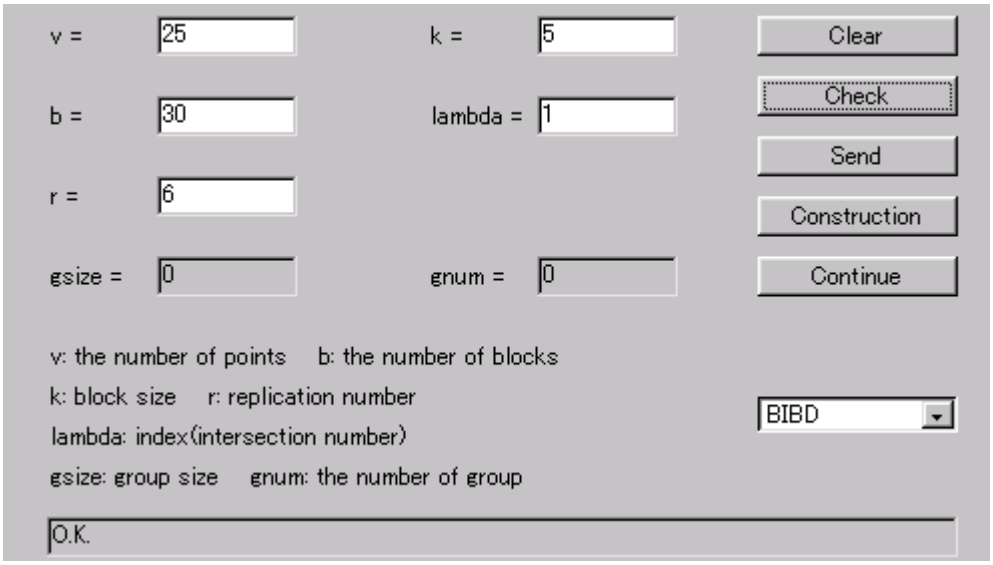

図 4: Step 4-1　パラメータのチェック 1 (条件式 (1.1) を満たすとき)

条件式 (1.1) が成り立たないときは, 'Such a design does not exist !!' が表示される. 図 5 はパラメータ v=50, k=3, r=28 を代入したときの様子である. '-1' はパラメータの値が求められない, もしくは自然数にならないことを示す.

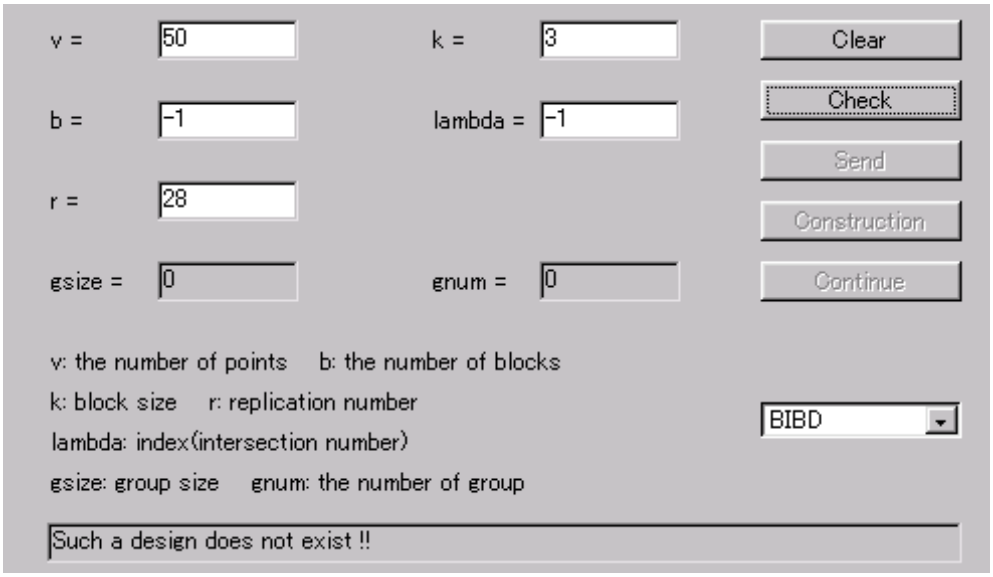

図 5: Step 4-2　パラメータのチェック 2 (条件式 (1.1) を満たさないとき)

また, BIB デザインについて v, k, λ の 3 つを入力したときは (1.1) 式が成り立たなくても, (1.1) 式が成り立つような最小の λ を次の計

算で求め, 表示する.

$$\lambda = \mathrm{lcm}\left(\frac{k-1}{\gcd\Big(k-1, v-1\Big)},\ \frac{k(k-1)}{\gcd\Big(k(k-1), v(v-1)\Big)}\right)$$

図 6, 7 はパラメータ v=42, k=6, λ=1 を代入したときの (1.1) 式が成り立つように, λ の値を補正した様子を表す.

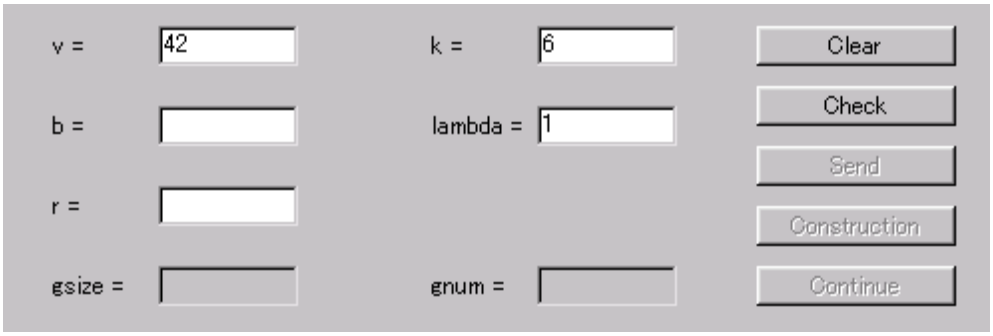

図 6: Step 4-3-1 λ の自動補正前

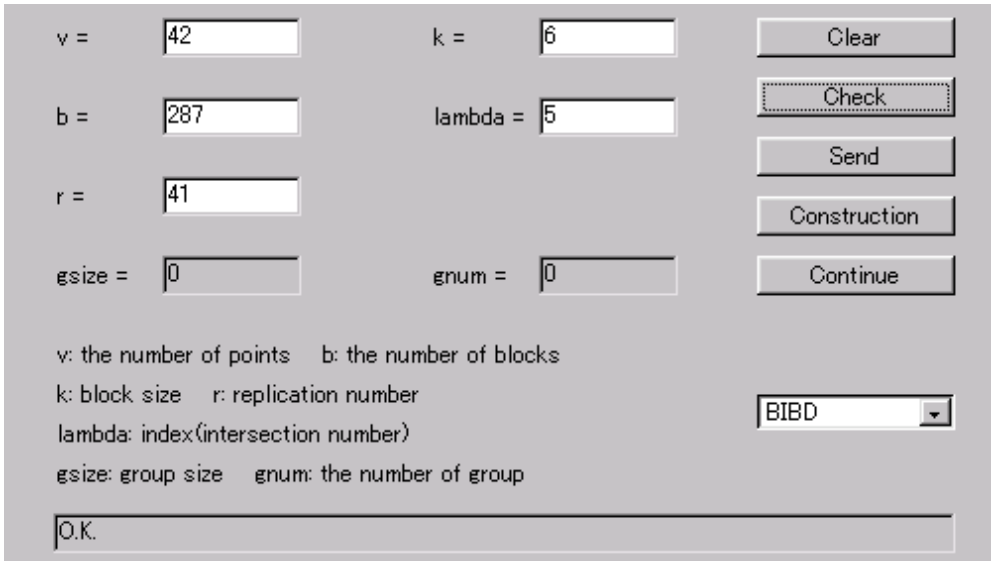

図 7: Step 4-3-2 λ の自動補正後

Step 5. 'OK' が表示されたら, 'send' ボタンを押す. するとパラメータがサーバに送られ, 探索が開始される. 探索が完了すると, テキストエリアに探索結果が表示される. 図 8 は $B[5, 1; 25]$ の存在を示す.

図 8: Step 5　　組合せデザインの探索結果 1

Step 6. BIB デザイン, group divisible デザインが存在するときは, パラメータを変えずに 'continue' ボタンを押すと, 他の構成法によりデザインが構成されるかどうかを探索し, 結果をテキストエリアに表示する. 見つからない場合は, 'Search of $B[k, \lambda; v]$ finished' のように表示される. 図 9 は図 8 の後, 4 回 'continue' ボタンを押した結果を表す.

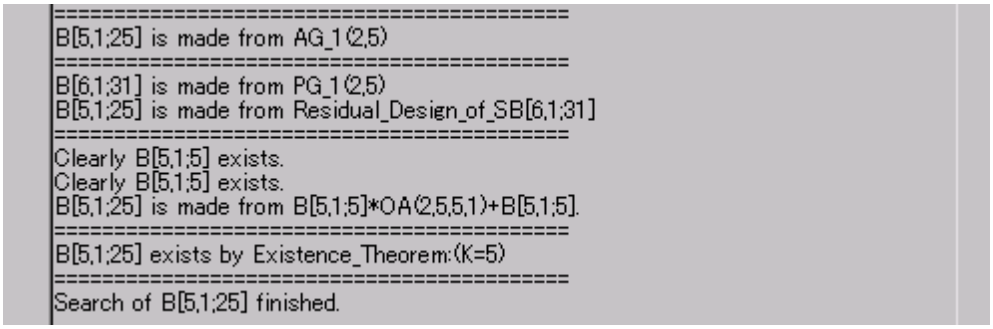

図 9: Step 6　　組合せデザインの探索結果 2

Step 7. BIB デザインについて, 射影幾何, アフィン幾何, アダマールデザイン, cyclic difference family, cyclic difference set により構成できる場合, 'construction' ボタンを押すとそのデザインの base ブロックを表示する. 図 10 は BIB デザイン $B[5, 1; 25]$ の base ブロックが, 位数 5 の有限体上の 2 次元アフィン幾何 $AG(2, 5)$ の 1 次元空間の集まり (1-flat) であることを意味している. $(\infty\ 0\ 6\ 12\ 18)$, $(1\ 10\ 14\ 15\ 17)$ はそれぞれ, Galois 体 $GF(5^2)$ 上の原始元 α を用いた 1-flat $(\alpha^{\infty}\,\alpha^{0}\,\alpha^{6}\,\alpha^{12}\,\alpha^{18})$, $(\alpha^{1}\,\alpha^{10}\,\alpha^{14}\,\alpha^{15}\,\alpha^{17})$ を指数部分のみで表し, これらを　mod 24 で cyclic に構成することでブロック全体を構成できる. ここで, α^{∞} は原点を表す.

図 10: Step 7　　BIB デザインの構成結果

3.2 DesNavi の動作例

いくつか例を挙げて, DesNavi の出入力結果の見方を説明する.

例 2 $B[9,1;5913]$ を探索する. このとき, b=485523, r=739 となる. DesNavi のサーバからは

```
B[9,1;81] is made from AG_1(2,9)
B[9,1;73] is made from PG_1(2,8)
B[9,1;5913] is made from B[9,1;81]*OA(2,9,73,1)+B[9,1;73].
```

というメッセージが返ってきて, アフィン幾何から作られる $B[9,1;81]$ と射影幾何から作られる $B[9,1;73]$, そして $OA(2,9,73,1)$ がそれぞれ存在するので, 逐次構成法により存在することがわかる. そこで, 'continue' ボタンを押してさらに探索すると,

```
============================================================
B[9,1;5913] exists by Existence_Theorem:(K=9)
============================================================
Search of B[9,1;5913] finished.
```

と, ブロックサイズ k に関する定理からもその存在を知ることができる.

例 3 $B[12,2;67]$ を探索する. このとき, b=67, r=12 となり, この *BIB* デザインは symmetric であることがわかる. この *BIB* デザインは非存在定理より, 存在しないことがわかる. このとき, サーバからは

```
B[12,2;67] does not exist from Symmetric_design(V:Odd):
                                    Bruck&Ryser&Chowla(BRC)(1949)
```

というメッセージが返ってくる.

例 4 $PD[5,5;144]$ の最大ブロック数を計算する.

```
The existence of an optimal PD[5,5;144] is unknown.
The maximum number of blocks of a PD[5,5;144] is at most 5126.
The optimal PD[5,4;144] exists.
The number of blocks of an optimal PD[5,4;144] is 4118.
Assaf and Hartman (1989)
Clearly B[5,1;5] exists.
B[5,1;145] exists by Existence_Theorem:(K=5)
B[5,1;145] exists.
Remove one point from B[5,1;145].
Then the optimal PD[5,1;144] exists.
The number of blocks of an optimal PD[5,1;144] is 1008.
Mills and Mullin (1992)
The maximum number of blocks of a PD[5,5;144] is at least 5126.
The maximum number of blocks of a PD[5,5;144] is 5126.
A PD[5,5;144] is optimal.
```

サーバから返ってきたメッセージの 1 行目により, $PD[5,5;144]$ が optimal であるか否かは不明であるが, (1.2) 式, Johnson Bound によって最大ブロック数の上限が 5126 であることがわかる. さらに, optimal $PD[5,4;144]$ と optimal $PD[5,1;144]$ を組み合わせることにより, $PD[5,5;144]$ を構成し, そのときのブロック数を求めると 5126 である. よって, $PD[5,5;144]$ の最大ブロック数は少なくとも 5126 であることがわる. つまり, $PD[5,5;144]$ の最大ブロック数は 5126 であり, optimal であることがわかる.

(注意) 一般に, optimal packing デザインを組み合わせてできる packing デザインであるとは限らない.

例 5 *GD* デザインを用いた *BIB* デザインの構成法の例を挙げる. $B[4,1;256]$ について探索を行う.

```
================================================
B[4,1;256] is made from AG_1(4,4)
================================================
B[4,1;256] exists by Existence_Theorem:(K=4)
================================================
Clearly B[4,1;4] exists.
B[4,1;256] is made from GDD[4,1,4;256]+B[4,1;4]
================================================
Clearly B[4,1;4] exists.
B[4,1;256] is made from GDD[4,1,3;255]+B[4,1;4]
================================================
Search of B[4,1;256] finished.
```

よって, グループサイズ 4, グループ数 64 の $GDD[4,1,4;256]$ と $B[4,1;4]$ を組み合わせて構成できる. さらに, グループサイズ 3, グループ数 85 の $GDD[4,1,3;255]$ と $B[4,1;4]$ を組み合わせても構成できることがわかる.

例 6 *BIB* デザインの実際のブロックの構成例を挙げる. $B[14,1;183]$ について探索を行う.

```
PG_1(2,13) is B[14,1;183].
The base blocks :
( 0 1 3 16 23 28 42 76 82 86 119 137 154 175)
mod 183
```

位数 13 の有限体上の 2 次元射影幾何 $PG(2, 13)$ の 1 次元空間の集まり (1-flat) が $B[14, 1; 183]$ のブロックとなる．DesNavi のサーバから返ってきた結果には, そのブロックの base ブロックが出力される.