

# Constructing some PBIBD(2)s by Tabu Search Algorithm

Luis B. Morales

IIMAS, Universidad Nacional Autónoma de México  
Apdo. Postal 70-221, México, D.F., 04510, Mexico  
lbm@servidor.unam.mx

## Abstract

Some papers have dealt with the construction of 2-concurrence designs as an optimization problem. In this paper we generalize these results to formulate also the construction of partially balanced incomplete block designs with 2 associate classes as a combinatorial optimization problem. We will propose an algorithm based on tabu search to construct many of these designs. Our algorithm was able to construct 51 resolvable and 71 non-resolvable PBIBD(2)s.

## 1 Introduction

The construction of incomplete block (IB) designs has been and remains a very active research area in discrete mathematics and statistics. Many algorithms have been given to construct certain types of designs, for example: partially balanced incomplete block designs with  $m$  associate classes (PBIBD( $m$ )s) [26], Steiner triples systems [24],  $\alpha$ -lattices [21] and some balanced incomplete block designs (BIBDs) [10]. However, there is no efficient algorithm for this purpose. Furthermore, there are trivial necessary conditions, but the sufficient conditions for the existence of designs with given parameters are unknown.

Though BIBDs can be constructed for a series of particular parameters, the general solution is unknown so far. These designs are known to exist for a limited number of parameter situations. Bose and Nair [3] defined partially balanced incomplete block designs with  $m$  associate classes (PBIBD( $m$ )s). These designs are used for example in plant breeding work [14] and group testing [25].

The construction or the existence and the resolvability of PBIBD( $m$ )s has received less attention than the balanced case. Clatworthy [6] has given

a list of partially balanced incomplete block designs with two associate classes. Since there are many parameter situations for which PBIBD( $m$ )s do not exist, Jarret [12] weakened the conditions and defined  $m$ -concurrency designs. Just as the balanced case, the general existence of  $m$ -concurrency and PBIBD( $m$ )s still remains as an open problem.

Several approaches have been proposed to construct incomplete block designs. Of these, the least explored and most interesting to us is the combinatorial optimization approach. In order to construct designs some optimization techniques have been applied: A hill-climbing algorithm has been used in the construction of Steiner triple designs and other combinatorial designs [24]. A simulated annealing algorithm was used by Mathon [13] to find non-isomorphic BIBDs. Elenbogen and Maxin [8] tried unsuccessfully to solve one instance of a resolvable regular 2-concurrency design using some techniques of combinatorial optimization. Recently a hill-climbing technique [11], a simple tabu search (TS) procedure [15] and a genetic algorithm [1, 23] could construct this design. Also a simple tabu search procedure was used in [16] and [27] to construct some resolvable regular 2-concurrency designs. More recently, Morales [18] used also a TS to construct many non-regular 2-concurrency designs. Also this author has constructed some difference families and eight new BIBDs in an optimization approach [19, 20].

In this paper the construction of resolvable and non-resolvable PBIBD(2) designs is formulated as a combinatorial optimization problem, in which symbols are assigned to blocks, trying to minimize the number of violations of some defining conditions. When the number of violations reaches zero, a valid design has been found. Even though time complexity for the general construction of block designs has not been analyzed, the completion of partial BIBD is shown to be an NP-complete problem, see [7]. In this paper we shall describe a heuristic procedure based on the tabu search technique to solve the combinatorial optimization problem for various instances of PBIBD(2)s.

This work, to our knowledge, represents the first effort to apply an optimization approach to construct PBIBD(2)s. For some design parameters the algorithm reached the optimal solution at 100% of the runs. But for other design parameters, our implementation could not produce an "optimal" solution. Tabu search could also find many PBIBD(2)s that were obtained before by other methods.

Section 2 gives the definition of PBIBD( $m$ )s, and some of their properties. We also define three matrices, which allow to formulate the construction of the designs in an optimization approach. In Section 3, we formulate the construction of these designs as a discrete optimization problem. In Section 4, a brief review of the basic principles of TS is given. Then, Section 5 presents an implementation of tabu search for the optimization

problem. Computational results are reported in Section 6, and the latter section consists of conclusions.

## 2 PBIBD( $m$ )s

In this section  $m$ -concurrency and PBIBD( $m$ ) are defined, as well as three matrices associated with them. Though the first two matrices are usually defined on any study of designs, here, we will use the matrices to formulate the construction of these designs as an optimization problem.

**Definition 1** *An equi-replicate incomplete block design (EIBD) is a pair  $(V, \mathcal{B})$  where  $V$  is a  $v$ -set of symbols and  $\mathcal{B}$  is a collection of  $b$   $k$ -subsets of  $V$  called blocks, such that each symbol occurs in  $r$  blocks, where  $k < v$ .*

**Definition 2** *An  $m$ -concurrency design is an equi-replicate incomplete block design satisfying the following conditions:*

- (1) *Any two symbols are either 1st, 2nd, ..., or  $m$ -th associates, the relation of association is symmetrical.*
- (2) *Each symbol  $\alpha$  has  $n_i$   $i$ -th associates ( $i = 1, \dots, m$ ), the numbers  $n_i$  being independent of the symbol  $\alpha$ .*
- (3) *Two symbols which are  $i$ -th associates occur together in exactly  $\lambda_i$  blocks ( $i = 1, \dots, m$ ), where  $\lambda_i$ 's are distinct.*

**Definition 3** *A partially balanced incomplete design with  $m$  associate classes is an  $m$ -concurrency design which satisfies the condition:*

- (4) *If any two symbols  $\alpha$  and  $\beta$  are  $i$ -th associates, then the number  $p_{jt}^i(\alpha, \beta)$  of symbols that are  $j$ -th associates of  $\alpha$  and  $t$ -th associates of  $\beta$  depends only on  $i, j, t$ , and not on  $\alpha$  and  $\beta$  (so we write this number as  $p_{jt}^i$ ).*

The numbers  $v, b, r, k, \lambda_i, n_i$  ( $i = 1, \dots, m$ ) are called the parameters of the first kind, whereas the numbers  $p_{jt}^i$  ( $i, j, t = 1, \dots, m$ ) are called the parameters of the second kind of the design.

Bose and Clatworthy in [5] showed that for partially balanced designs with two associate classes, it is unnecessary to assume the constancy of all the parameters  $p_{jt}^i$  because, if  $p_{11}^1$  and  $p_{11}^2$  are constant, then all parameters of the second kind are also constant. This important property will be used to simplify the construction of PBIBD(2)s.

Let us now define the matrices associated with the designs. With an equi-replicate incomplete block design we associate a  $v \times b$  matrix  $A = (a_{ij})$  where

$$a_{ij} = \begin{cases} 1, & \text{if symbol } i \text{ belongs to block } j, \\ 0, & \text{otherwise.} \end{cases}$$

We call  $A$  the *incidence* matrix of the design. Clearly, this matrix satisfies

$$\sum_{i=1}^v a_{ij} = k, \quad 1 \leq j \leq b, \quad (1)$$

$$\sum_{j=1}^b a_{ij} = r, \quad 1 \leq i \leq v. \quad (2)$$

Conversely, if  $A$  is a  $v \times b$  matrix of zeros and ones satisfying (1) and (2), it can easily be verified that  $A$  is the incidence matrix of an EIBD.

If  $A$  is the incidence matrix of an equi-replicate incomplete block design, then the *concurrency* matrix of the design is the symmetrical  $v \times v$  matrix  $B = AA'$  ( $A'$  being the transpose of  $A$ ). Clearly, for every two distinct symbols  $\alpha$  and  $\beta$ , the entry  $b_{\alpha\beta}$  is the number of blocks where they occur together. Note that the matrix  $B$  has  $r$  on the diagonal. It follows from Definition 2.2 that each row of the concurrency matrix  $B$  of an  $m$ -concurrency design consists of

$$\overbrace{\lambda_1, \dots, \lambda_1}^{n_1}, \overbrace{\lambda_2, \dots, \lambda_2}^{n_2}, \dots, \overbrace{\lambda_m, \dots, \lambda_m}^{n_m} \quad \text{and } r \text{ on the diagonal.} \quad (3)$$

Conversely, if  $A$  is a  $v \times b$  matrix of zeros and ones satisfying (1) and (2) and the matrix  $B = AA'$  satisfies (3), then, it is not hard to verify that  $A$  and  $B$  are respectively the incidence and concurrency matrices of an  $m$ -concurrency design with the parameters  $v, b, r, k, \lambda_i$  and  $n_i$  ( $i = 1, \dots, m$ ).

The last matrix we will define is the intersection matrix  $C = (c_{\alpha\beta})$  of order  $v \times v$  associated to an equi-replicate incomplete block design, where  $c_{\alpha\beta}$  is the number of symbols that are first associate to  $\alpha$  and also first associate to  $\beta$ , for  $\alpha \neq \beta$ . (Two symbols are first associates, if they occur together in exactly  $\lambda_1$  blocks). The diagonal is undefined. It is not hard to see from the definition of PBIBD(2)s that this matrix is symmetric, and for any two different symbols  $\alpha$  and  $\beta$

$$c_{\alpha\beta} = \begin{cases} p_{11}^1, & \text{if symbols } \alpha \text{ and } \beta \text{ are 1st associates,} \\ p_{11}^2, & \text{if symbols } \alpha \text{ and } \beta \text{ are 2nd associates.} \end{cases} \quad (4)$$

In any 2-concurrency design the constancy of the parameters  $p_{11}^1$  and  $p_{11}^2$  imply the constancy of the all parameters of the second kind [5]. Thus, the

matrices  $A$ ,  $B = AA'$  and  $C$  are respectively the incidence, concurrence and intersection matrices of a partially balanced incomplete block design with two associate classes with parameters  $v, b, r, k, \lambda_i, n_i, p_{jt}^1, p_{jt}^2$  ( $i, j, t = 1, 2$ ) if and only if  $A$  is a  $v \times b$  matrix of zeros and ones that satisfies (1) and (2),  $B$  satisfies (3) and  $C$  is a  $v \times v$  matrix satisfying (4).

It is well known that necessary conditions for the existence of a PBIBD( $m$ ) with the parameters  $v, b, r, k, \lambda_i, n_i, p_{jt}^i$  ( $i, j, t = 1, \dots, m$ ) are

$$vr = bk, \quad (5)$$

$$\sum_{i=1}^m n_i = v - 1, \quad (6)$$

$$r(k-1) = \sum_{i=1}^m n_i \lambda_i, \quad (7)$$

$$\sum_{t=1}^m p_{jt}^i = n_j - \delta_{ij}, \quad (8)$$

$$n_i p_{jt}^i = n_j p_{it}^j. \quad (9)$$

If the blocks of a design can be divided into  $r$  subsets  $P_i$  called *parallel classes* of size  $q = b/r$  so that every symbol of  $V$  appears in one block from each class, the design is called *resolvable*. From this definition, we have a further necessary condition for the existence of a resolvable design:  $k$  must divide  $v$ .

Balanced incomplete block designs are a special case within  $m$ -concurrence designs when  $m = 1$ . Since these designs have only one associate class,  $n_1 = v - 1$ , so  $p_{11}^1 = v - 1$ . For any balanced incomplete block design, the entries of its concurrence matrix  $B$  are  $r$  on the diagonal and  $\lambda$  elsewhere. Notice that for  $m = 1$ , the 1-concurrence designs and the partially balanced incomplete designs with 1 associate classes are the same.

Other simple class of partially balanced incomplete designs, apart from the balanced incomplete block designs, are those with  $m = 2$  distinct concurrences. For these designs,  $n_1, n_2 > 0$  and  $\lambda_1 \neq \lambda_2$ . If  $\lambda_1$  is the integer part of  $r(k-1)/(v-1)$  and  $\lambda_1 = \lambda_2 = 1$  the design is called a *regular graph design*.

The concept of an associate class was developed by Bose and Shimamoto [4] and it is the basis for a classification of PBIBD(2)s. These designs are classed into six types, namely: group divisible, triangular, Latin square types, cyclic, partial geometry and miscellaneous.

### 3 Optimization

As observed in the previous section, for constructing an  $m$ -concurrence design it is necessary to find a matrix  $A$  of zeros and ones that satisfies (1) and (2), and the matrix  $B = AA'$  must satisfy (3). Although it is not hard to find a matrix  $A$  that satisfies the first two constraints, the last constraint can be very difficult to satisfy in practice. Thus, the construction of EIBDs is trivial, but the construction of  $m$ -concurrence designs and PBIBD( $m$ )s is not an easy problem.

Given two integers  $t$  and  $s$  we define the function

$$H_{t,s}(x) = \begin{cases} (x-t)^2, & \text{if } x \neq s, \\ 1, & \text{if } x = s. \end{cases}$$

Let  $v, b, r, k, \lambda_1, \lambda_2, n_1$  and  $n_2$  be integers satisfying (5)-(7), and  $\lambda_1 < \lambda_2$ . Let  $D$  be an EIBD with parameters  $v, b, r$  and  $k$ . Then, the  $(\lambda_1, \lambda_2)$ -imbalance of  $D$  is defined as

$$\sigma_{\lambda_1, \lambda_2}(D) = \sum_{\alpha < \beta} H_{\lambda_1, \lambda_2}(b_{\alpha\beta}), \quad (10)$$

where  $(b_{\alpha\beta})$  is the concurrence matrix of the design  $D$ . Morales [17] proved that for any EIBD  $D$  with parameters  $v, b, r$  and  $k$ , we have  $\sigma_{\lambda_1, \lambda_2}(D) \geq vn_2/2$  and this value is attained if and only if  $D$  is a 2-concurrence design with parameters  $v, b, r, k, \lambda_1, \lambda_2, n_1$  and  $n_2$ . So, he formulated the construction of 2-concurrence designs as a minimization problem with the cost function  $\sigma_{\lambda_1, \lambda_2}$ . Before, Brown [2] proved a similar result for regular 2-concurrence designs.

However, to include the constancy of the parameters  $p_{11}^1$  and  $p_{11}^2$  in an optimal way, we must use the intersection matrix  $C = (c_{\alpha\beta})$ . A  $v \times b$  matrix  $A$  of zeros and ones defines a PBIBD with two associate classes if the matrix  $A$  satisfies constraints (1), (2), and the matrix  $B = AA'$  satisfies (3), and further the matrix  $C$  satisfies (4). Therefore, to formulate the construction of PBIBD(2)s as an optimization problem, it is necessary to add a new component to the cost function. This function  $g$  computes the degree of violations of the constraint (4). Hence, the second component of the cost function is

$$g(C) = \sum_{\alpha < \beta} Q(c_{\alpha\beta}),$$

where

$$Q(c_{\alpha\beta}) = \begin{cases} 0, & \text{if } c_{\alpha\beta} = p_{11}^2 \text{ and } \alpha \text{ and } \beta \text{ are second associate,} \\ (c_{\alpha\beta} - p_{11}^1)^2, & \text{otherwise.} \end{cases}$$

Clearly, the lower bound of  $g$  is zero. The intersection matrix  $C$  of a 2-concurrence design satisfies (4) if and only if  $g(C) = 0$ , because any pair of symbols is either first or second associate. Therefore, a partially balanced incomplete block design with two associate classes is constructed whenever the cost function  $\sigma_{\lambda_1, \lambda_2} + g$  reaches the optimal value  $vn_2/2$  and the matrix  $A$  of zeros and ones satisfies constraints (1) and (2). Thus, the construction of PBIBD(2)s can also be formulated as an optimization problem, but with the cost function  $\sigma_{\lambda_1, \lambda_2} + g$ .

For the group divisible, triangular, Latin square types, cyclic and partial geometry PBIBD(2)s, the parameters  $p_{11}^1$  and  $p_{11}^2$  can be calculated from the parameters of the first class (see [6]). While for the miscellaneous designs these parameters cannot be known beforehand. Nevertheless, there are bounds for the parameters (see [12, Theorems 3.2-3.3]). Using these bounds, (8) and (9) we obtain all possible values for these parameters. Then, each one of these values is taken in the optimization process until a PBIBD(2) is found.

In order to formulate the construction of resolvable designs as an optimization problem, we need another constraint for the matrix  $A$ :

$$\sum_{B_j \in P_i} a_{\alpha j} = 1 \text{ for each symbol } \alpha \text{ and for each parallel class } P_i, i = 1, \dots, r. \quad (11)$$

Clearly, the incidence matrix of an EIBD satisfies (11) if and only if the design is resolvable. Adding the corresponding constraint (11) to the preceding optimization problems, we can also formulate the construction of resolvable PBIBD(2)s as an optimization problem.

## 4 Tabu Search Heuristic

The tabu search method is an iterative heuristic method used for finding, in a set  $X$  of feasible solutions, the solution that minimizes an objective function  $f$  based on neighborhood search (NS).

In neighborhood search, each feasible solution  $x$  has an associated set of neighbors,  $N(x) \subset X$ , called the *neighborhood* of  $x$ . NS starts from an initial feasible solution chosen at random and explores the space  $X$  by moving from one solution to another one in its neighborhood. At each iteration of the process, a subset  $V$  of  $N(x)$  is generated and we move from the current solution  $x$  to the best one  $x^*$  in  $V$ , whether or not  $f(x^*)$  is better than  $f(x)$ . If  $N(x)$  is small, it is possible to take  $V$  as the entire neighborhood.

In some optimization problems, including the problem at hand, when exploring the set  $V$  we find that there are multiple optimal solutions. Then,

an important feature in these problems is to make a random choice of the best solution  $x^*$  of  $V$ . As we will see in the next section, this allows one to obtain the theoretical minimum value  $vn_2/2$  for many parameter situations. Nevertheless, the main shortcoming of this simple heuristic optimization algorithm is a cycling problem.

Stopping rules must also be defined; in many cases (including our optimization problem) a lower bound  $f^*$  of the objective function is known in advance. As soon as we have reached this bound, we may interrupt the algorithm. In general, however,  $f^*$  is not available with sufficient accuracy, so a fixed maximum number of iterations is given.

The tabu search algorithm offers another interesting possibility for overcoming the above-mentioned obstacle of the NS technique. To prevent cycling, a queue called the *tabu list*  $T$  of length  $|T| = t$  is provided. Its aim is to forbid moves between solutions that reinstate certain attributes of past solutions. After  $t$  iterations they are removed from the list and are free to be reinstated. The tabu list is also called a *short term memory function*, because it stores information on the  $t$  most recent moves. Likewise, a *long-term memory* is used to improve the search process as we shall see in the adaptation of this technique. This memory is used to diversify the search and move to unexplored regions, and is usually based on a frequency criteria.

Unfortunately, the tabu list may forbid certain interesting moves, such as those that lead to a better solution than the best one found so far. An *aspiration criterion* is introduced to cancel the tabu status of a move when this move is judged useful.

Note that neighborhood search is a tabu search method without an aspiration function and where the length of the tabu list is zero.

## 5 Adaptation of Tabu Search

The most important points in the implementation of TS for our particular application are: the search space  $X$ , the objective function  $f$ , the neighborhood function  $N(x)$ , the method of choosing the initial solution, the length of the tabu list  $T$ , the aspiration criterion and the stop criterion.

For the construction of resolvable and non-resolvable PBIBD(2)s, we define the search space  $X$  as the family of all resolvable and non-resolvable EIBDs, respectively.

Two EIBDs  $D'$  and  $D$  are defined as neighbors if one of them is obtained from the other by exchanging two symbols  $i$  and  $j$  from different blocks  $B_l$  and  $B_m$ . For resolvable designs, the blocks  $B_l$  and  $B_m$  must be chosen from the same parallel class. Note that the move that transforms  $D$  into  $D'$  is defined by the four-dimension vector  $(i, j, l, m)$ . In terms of the incidence

matrix,  $X'$  is a neighboring solution of  $X$  if for some  $i, j, l, m$ :

$$\begin{aligned} x'_{il} &= 0 \quad \text{if } x_{im} = 1, & x'_{im} &= 1 \quad \text{if } x_{il} = 0, \\ x'_{jl} &= 1 \quad \text{if } x_{jm} = 0, & x'_{jm} &= 0 \quad \text{if } x_{jl} = 1, \\ x'_{pq} &= x_{pq}, & \text{for } (p, q) &\neq (i, l), (j, m), (j, l), (i, m). \end{aligned}$$

It is not hard to see that each resolvable design has  $r \binom{g}{2} k k$  resolvable neighbors [8]. However, for non-resolvable designs the number of neighbors varies with respect to the design configuration. Since these designs allow the exchange of two symbols from any two different blocks (not necessarily in the same parallel class), the number of non-resolvable neighbors is greater than those resolvable. In the construction of resolvable designs, we take  $V = N(x)$ . On the contrary, for non-resolvable designs we generate randomly a subset  $V \subset N(x)$  with size  $0.8|N(x)|$ .

The objective function used was  $\sigma_{\lambda_1, \lambda_2} + g$  (defined in section 3). The definition of the neighborhood and the objective function allow one to calculate the change in the function  $f$  without recomputing the cost of the entire design. After the move  $(i, j, l, m)$ , the only entries of the concurrence matrix  $C$  that change are those belonging to the  $i$ -th or  $j$ -th rows, or columns. Therefore, the value of a move can be calculated in  $O(v)$  operations. By contrast, the entire matrix  $C$  can be calculated in  $O(v^2)$ , because the order of  $C$  is  $v$ . Note that the complexity has been reduced by a factor of  $1/v$ .

An initial resolvable EIBD is generated as follows: In each parallel class we randomly split  $v$  symbols into  $g$  blocks, each one of them with  $k$  symbols. However, the method of choosing random initial solutions for non-resolvable designs is most difficult. A generating procedure for resolvable designs may be written in pseudo-code as follows:

```

c[i] = 0, x[i, j] = 0, 1 ≤ i ≤ v, 1 ≤ j < b
for j = 1 to b do
begin
  n = 1
  choose the symbols 1, ..., v in index order randomly e[1], ..., e[v]
  sort the sequence e[1], ..., e[v] such that c[e[i]] ≤ c[e[i + 1]], 1 ≤ i < b
  l = 1
  while (n ≤ k) do
  begin
    if (c[e[l]] < r) then
      begin
        x[e[l], j] = 1
        c[e[l]] ++
        n ++

```

```

    end
  l++
end
end

```

To construct the tabu list, we forbid, during  $t$  iterations, the exchange of any two symbols that were exchanged between two different blocks. Formally, the tabu list consists of vectors  $(i, j, l, m)$ , where  $i$  and  $j$  are symbols that cannot be exchanged from blocks  $B_l$  and  $B_m$ . Since for resolvable designs the blocks in each parallel class are determined by any of their symbols, then the tabu list –for resolvable designs– consists of triples  $(i, j, l)$ , where  $i$  and  $j$  are symbols that cannot be exchanged in the parallel class  $l$ .

The long term memory is a function that records moves taken in the past in order to penalize those which are non-improving. The goal is to diversify the search by compelling regions to be visited that possibly were not explored before [9]. In our particular TS implementation, the long-term memory is a  $v \times v$  matrix, which will be denoted  $F$ . The matrix has zeroes at the beginning of the procedure. When a pair of symbols  $(s, t)$  are swapped at a given iteration, the matrix changes as follows:  $F_{st} = F_{st} + 1$ . (The entry  $(F_{s,t})$  is the frequency at which the symbols  $s$  and  $t$  have been swapped). Then, the values of non-improving moves that switch the symbols  $s$  and  $t$  are increased by  $F_{st}$ .

We used the standard aspiration criterion. This criterion allows the tabu status of a move from  $D$  to  $D'$  to be canceled if the value  $f(D')$  is strictly better than the best value obtained so far. This means that the tabu status of a move from  $D$  to  $D'$  may be dropped if  $f(D') < f(D^o)$ , where  $D^o$  is the best solution found so far.

The process stops if the function  $f$  has reached the global minimum  $vn_2/2$ . However, since TS is a heuristic technique, it does not always guarantee reaching an optimal solution, and the search process will be stopped if the number of iterations used without improving the best solution is greater than a *nimax* limit.

## 6 Numerical Results

The TS algorithms described above were implemented in C language. They are available for free downloading from the website

<http://www.mcc.unam.mx/~lbn>

Our heuristic algorithms were used to construct resolvable and nonresolvable PBIBD(2)s on the set problems of designs with parameters  $v \leq b/2$ ,  $8 \leq v \leq 24$ ,  $5 \leq r \leq 20$  and  $3 \leq k \leq 8$ . All tabu search runs were carried

out with a random initial solution. TS procedure was carried out 20 times using  $nimax = 900$  on each instance tested.

In our experiments, we have observed that there are instances very easy to solve (always in less than 100 iterations), whilst others are difficult, but TS obtained, with high probability (up to 90%), an optimal solution with an average number of iterations ranging from 150 to 400; other instances are very hard. For later instances, it was necessary to incorporate a long memory term -frequency matrix- to obtain an optimal solution, and our algorithm never reached 90% of successful runs. Unfortunately, there were many design parameters where our heuristic methods produced no optimal solution. Our algorithms were able to construct 51 resolvable and 71 nonresolvable PBIBD(2)s. Table 1 presents the list of design parameters of PBIBD(2)s that TS was able to produce. Each row of this table corresponds to an instance. Columns 2-8 show the parameters of the first kind. Columns 9-10 give respectively the parameters  $p_{11}^1$  and  $p_{11}^2$  (the other six parameters of the second kind can be calculated from equations (8) and (9)). Column ITO gives the average number of iterations where the global minimum was found. Column POS shows the percentage of runs where the global minimum was found. Column AS indicates the association scheme for the design in the Clatworthy classification [6].  $GD(n, m)$  indicates that the association scheme for the design is a group divisible with parameters  $m$  and  $n$ .  $T(n)$  indicates that the PBIBD(2) is a triangular design with parameter  $n$ .  $L_i(n)$  indicates that the association scheme for the design is of latin square type with parameters  $n$  and  $i(\geq 2)$  constraints.  $GD^*(m, n)$  and  $T^*(n)$  indicate that in the designs  $GD(n, m)$  and  $T(n)$  the label 1st and 2nd associates have been swapped.

We have tested the random neighborhood search and tabu search on the problems at hand. The impact of the length tabu was investigated. For easy instances, the difference between NS and TS is not significant, both give always the global minimum at equivalent CPU time. For difficult and hard instances TS gives the best results. The results for these instances show that the best length tabu seems to be an integer somewhere between 4 and 7.

In our experiments on difficult and hard instances, we have seen that for a fixed problem, the number of iterations needed to find an optimal solution strongly depends on the initial solution. For example, a PBIBD(2) with parameters (10, 20, 12, 6, 7, 4, 8, 1, 6, 8) was found in 8 iterations with one initial solution, while about 903 iterations were required to reach another optimal solution using a different initial solution.

Finally, Table 2 gives a resolvable PBIBD(2) with parameters  $v = 12$ ,  $b = 30$ ,  $r = 10$ ,  $k = 4$ ,  $\lambda_1 = 2$ ,  $\lambda_2 = 3$ ,  $n_1 = 3$ ,  $n_2 = 8$ ,  $p_{11}^1 = 2$  and  $p_{11}^2 = 2$  obtained by TS algorithm. Each row in Table 2 corresponds to a parallel class of the design. Also, in this table we give the concurrence and

Table 2. A (12,30,10,4,2,3,3,8,2,0)-RPIBD(2) and its concurrence and intersection matrices obtained by TS

2 6 7 10	3 8 9 12	1 4 5 11
1 3 9 10	5 6 7 8	2 4 11 12
3 8 10 11	4 5 7 9	1 2 6 12
2 6 9 11	3 4 5 7	1 8 10 12
1 2 3 5	4 6 10 11	7 8 9 12
2 4 8 9	5 6 10 12	1 3 7 11
1 4 9 10	2 7 8 11	3 5 6 12
1 6 7 9	2 3 4 12	5 8 10 11
5 9 11 12	2 3 7 10	1 4 6 8
2 5 9 10	1 7 11 12	3 4 6 8

concurrence matrix	Intersection matrix
0 2 3 3 2 3 3 2 3 3 3 3	* 2 0 0 2 0 0 2 0 0 2 0 0 0 0
2 0 3 3 2 3 3 2 3 3 3 3	2 * 0 0 2 0 0 2 0 0 2 0 0 0 0
3 3 0 3 3 2 3 3 2 3 2 3	0 0 * 0 0 2 0 0 2 0 2 0 2 0
3 3 3 0 3 3 2 3 3 2 3 2	0 0 0 * 0 0 2 0 0 2 0 2 0 2
2 2 3 3 0 3 3 2 3 3 3 3	2 2 0 0 * 0 0 2 0 0 0 0 0 0
3 3 2 3 3 0 3 3 2 3 2 3	0 0 2 0 0 * 0 0 2 0 2 0 2 0
3 3 3 2 3 3 0 3 3 2 3 2	0 0 0 2 0 0 * 0 0 2 0 2 0 2
2 2 3 3 2 3 3 0 3 3 3 3	2 2 0 0 2 0 0 * 0 0 0 0 0 0
3 3 2 3 3 2 3 3 0 3 2 3	0 0 2 0 0 2 0 0 * 0 0 2 0 0
3 3 3 2 3 3 2 3 3 0 3 2	0 0 0 2 0 0 2 0 0 * 0 0 2 0
3 3 2 3 3 2 3 3 2 3 0 3	0 0 2 0 0 2 0 0 2 0 0 * 0 2 0
3 3 3 2 3 3 2 3 3 2 3 0	0 0 0 2 0 0 2 0 0 2 0 0 * 0 2
3 3 3 2 3 3 2 3 3 2 3 0	0 0 2 0 0 2 0 0 2 0 0 2 0 * 0
3 3 3 2 3 3 2 3 3 2 3 0	0 0 0 2 0 0 2 0 0 2 0 0 2 0 *

intersection matrices of the design.

## 7 Conclusions

In this paper, we used a recent result on 2-concurrence designs to formulate the construction of PBIBD(2)s as a discrete optimization problem where we require optimal solutions rather than close approximations to them. The theoretical minimum of the objective function is known in advance, and this value is reached if and only if a PBIBD(2) is constructed. We have shown that it is possible to find an optimal solution to this problem using a procedure based on TS. An important feature incorporated into TS was to make a random choice whenever there are multiple best solutions. This

tends to diversify the search and helps preventing cycling.

From the computational experiments we have seen that there are design parameters which are easy, difficult and hard to construct, and others where TS produces no optimal solutions. In easy and difficult instances a tabu search procedure with one tabu list is sufficient to solve them. However, for other designs it was necessary to incorporate a long memory term—frequency matrix—to obtain an optimal solution. So, TS procedures construct some new PBIBD(2)s which are not available in the catalogues of experimental designs. Moreover TS was able to construct other designs whose existence already was known. The experiments on the difficult and hard instances have shown that for a fixed problem, the number of iterations needed to find an optimal solution strongly depends on the initial solution.

TS was able to construct 51 nonregular resolvable and 71 nonresolvable PBIBD(2)s, many of them with high probability (up to 90%) of successful runs. Unfortunately, TS did not produce optimal solutions for many of the tested problem instances.

## References

- [1] D. Ashlock. *Finding designs with genetic algorithms*, Computational and Constructive Design Theory, Ed. W.D. Wallis, Klumer Academic Pub. (1996), 49–65.
- [2] R.B. Brown, Nonexistence of a regular graph design with  $r = 17$  and  $k = 6$ . *Discrete Math.* **68** (1998), 315–318.
- [3] R. C. Bose and K. R. Nair, Partially balanced incomplete block designs, *Sankhya*, **4** (1939), 199–204.
- [4] R. C. Bose and T. Shimamoto, Classification and Analysis of partially balanced incomplete block Designs with two associate classes, *J. Amer. Statist. Assn.* **47** (1952), 151–184.
- [5] R. C. Bose and W. H. Clatworthy, Some classes of partially designs. *Ann. Math. Stat.* **26** (1955), 212–232.
- [6] W. H. Clatworthy, *Tables of two-associate-class partially balanced designs*. Nat. Bureau of Stand., Appl. Math. Series 63 (1973).
- [7] C. J. Colbourn, Embedding partial Steiner triple system is NP-complete, *J. of Combinat. Theory (A)* **35** (1983), 100–105.
- [8] B.S. Elenbogen and B.R. Maxim, (1992). Scheduling a bridge club (A case study in discrete optimization), *Math. Magazine* **65** (1992), 18–26.

- [9] F. Glover, Tabu search Part I, *ORSA Journal on Computing* **1** (1989), 190–206.
- [10] H. Hanani, Balanced incomplete designs and related designs, *Discrete Math* **11** (1975), 255–369.
- [11] D.L. Kreher, G.F. Royle and W.D. Wallis, A family of resolvable regular graph designs, *Discrete Math.* **156** (1996), 269–273.
- [12] R. G. Jarrett, Definitions and properties for  $m$ -concurrence designs, *J. R. Statis. Soc. B.* **45** (1983), 1–10.
- [13] R. Mathon, Nonisomorphic designs by simulated annealing, preprint.
- [14] O. Mayo, *The theory of plant breeding*, Clarendon Press, Oxford, 1987.
- [15] L.B. Morales, Scheduling a bridge club by tabu search, *Math. Magazine* **70** (1997), 287–290.
- [16] L.B. Morales and F. Maldonado, Constructing optimal schedules for certain types of tournaments using tabu search, *Investigación Operativa* **6** (1998), 127–135.
- [17] L.B. Morales, A theorem on 2-concurrence designs, *Journal of Combinatorial Math. and Combinatorial Computing* **34** (2000), 71–75.
- [18] L.B. Morales, Tabu search algorithm for the construction of nonregular 2-concurrence designs, preprint.
- [19] L.B. Morales, Constructing difference families in an optimization approach. Six new BIBDs, *Journal of Combinatorial designs* **8** (2000), 261–273.
- [20] L.B. Morales, Two new 1-rotational (36,9,8) and (40,10,9) RBIBDs, *Journal of Combinatorial Math. and Combinatorial Computing* **36** (2001) 119–126.
- [21] L. J. Patterson, H. D. Patterson, An algorithm for constructing  $\alpha$ -lattice designs, *Ars Combinatoria* **16A** (1988), 87–98.
- [22] D. Raghavarao, *Constructions and combinatorial problems in design of experiments*. J. Wiley & Sons, New York (1971).
- [23] R.J. Simpson, Scheduling a bridge club using a genetic algorithm, *Math. Magazine* **70** (1997), 281–286.
- [24] D. R. Stinson, Hill-Climbing, Algorithm for the construction of combinatorial designs, *Annals of Discrete Math.* **26** (1985), 321–334.

- [25] F. Vakil and M. Parnes, On the structure of a class of sets useful in nonadaptive group testing. *J. Stat. Plann. Infer.* **30** (1994), 57–69.
- [26] W. D. Wallis, A construction for partially balanced incomplete block designs. *The Austral. J. Statist.* **12** (1970), 160–161.
- [27] M. Zamudio, Búsqueda tabú para la construcción de diseños de experimentos, Master thesis, CCH-IIMAS, UNAM (1996).

Table 1. PBIBD(2)s obtained by TS

No.	v	b	r	k	$\lambda_1$	$\lambda_2$	$n_1$	$n_2$	$p_{11}^1$	$p_{11}^2$	ITO	POS	AS
1†	15	25	5	3	0	1	4	10	3	0	193	100	$GD(3, 5)$
2†	9	15	5	3	1	2	6	2	3	6	43	100	$GD^*(3, 3)$
3	12	20	5	3	0	1	1	10	0	0	10	100	$GD(6, 2)$
4	8	10	5	4	2	3	6	1	4	6	6	100	$GD^*(4, 2)$
5†	15	30	6	3	0	1	2	12	1	0	199	100	$GD(5, 3)$
6†	15	30	6	3	0	2	8	6	4	4	531	45	$T(6)$
7†	9	18	6	3	1	3	6	2	3	6	15	100	$GD^*(3, 3)$
8	10	20	6	3	1	4	8	1	6	8	426	85	$GD^*(5, 2)$
9†	9	18	6	3	2	0	6	2	3	6	840	10	$GD^*(3, 3)$
10	8	16	6	3	2	0	6	1	4	6	55	100	$GD^*(4, 2)$
11†	16	24	6	4	1	2	12	3	8	12	285	100	$GD^*(4, 4)$
12†	8	12	6	4	2	3	3	4	2	0	6	100	$GD(2, 4)$
13	9	18	6	3	1	3	6	2	3	6	7	20	$GD^*(3, 3)$
14	12	24	6	3	1	2	10	1	8	10	10	100	$GD^*(6, 2)$
15	9	18	6	3	2	0	6	2	3	6	497	15	$GD^*(3, 3)$
16	20	30	6	4	0	1	1	18	0	0	629	35	$GD(10, 2)$
17	12	18	6	4	2	0	9	2	6	9	82	25	$GD^*(4, 3)$
18	10	12	6	5	2	3	3	6	0	1	10	100	$T^*(5)$
19†	9	21	7	3	1	2	2	6	1	0	22	100	$GD(3, 3)$
20†	12	28	7	3	1	2	8	3	4	8	365	95	$GD^*(3, 4)$
21†	9	21	7	3	1	4	6	2	3	6	69	100	$GD^*(3, 3)$
22†	16	28	7	4	1	2	9	6	4	6	260	75	$L_3(4)$
23†	16	28	7	4	1	3	12	3	8	12	851	20	$GD^*(4, 4)$
24	12	21	7	4	1	2	1	10	0	0	22	100	$GD(6, 2)$
25	10	14	7	5	3	4	8	1	6	8	21	100	$GD^*(5, 2)$
26	12	14	7	6	3	5	10	1	8	10	107	100	$GD^*(6, 2)$
27†	12	32	8	3	1	2	6	5	0	6	382	95	$GD^*(2, 6)$
28†	15	40	8	3	1	2	12	2	9	12	581	10	$GD^*(5, 3)$
29†	16	32	8	4	1	2	6	9	2	2	588	20	$L_2(4)$
30	14	28	8	4	2	0	12	1	10	12	686	40	$GD^*(7, 2)$
31	10	20	8	4	2	4	6	3	3	4	696	15	$T(5)$
32	10	20	8	4	3	0	8	1	6	8	265	55	$GD^*(5, 2)$
33†	8	16	8	4	3	4	4	3	0	4	5	100	$GD^*(2, 4)$
34†	8	16	8	4	4	0	6	1	4	6	13	65	$GD^*(4, 2)$
35	18	48	8	3	0	1	1	16	0	0	26	100	$GD(9, 2)$
36	24	48	8	4	1	2	22	1	20	22	1005	25	$GD^*(12, 2)$
37	12	16	8	6	4	0	10	1	8	10	103	70	$GD^*(6, 2)$
38†	9	27	9	3	2	3	6	2	3	6	46	100	$GD^*(3, 3)$
39†	9	27	9	3	3	0	6	2	3	6	991	55	$GD^*(3, 3)$
40	8	24	9	3	3	0	6	1	4	6	25	85	$GD^*(4, 2)$
41†	16	36	9	4	1	2	3	12	2	0	354	20	$GD(4, 4)$

Note. Designs marked † are resolvable

Table 1. PBIBD(2)s obtained by TS – continued

No.	v	b	r	k	$\lambda_1$	$\lambda_2$	$n_1$	$n_2$	$p_{11}^1$	$p_{11}^2$	ITO	POS	AS
42†	12	27	9	4	2	3	6	5	0	6	227	100	$GD^*(2, 6)$
43†	12	27	9	4	3	0	9	2	6	9	685	15	$GD^*(4, 3)$
44†	18	27	9	6	3	0	15	2	12	15	1145	10	$GD^*(6, 3)$
45	8	18	9	4	3	4	1	6	0	0	12	100	$GD(4, 2)$
46	12	18	9	6	4	5	10	1	8	10	111	100	$GD^*(6, 2)$
47†	12	40	10	3	1	2	2	9	1	0	473	15	$GD(4, 3)$
48†	12	40	10	3	2	0	10	1	8	10	406	15	$GD^*(6, 2)$
49†	9	30	10	3	2	4	6	2	3	6	124	100	$GD^*(3, 3)$
50†	9	30	10	3	3	1	6	2	3	6	706	20	$GD^*(3, 3)$
51†	12	30	10	4	2	3	3	8	2	0	540	5	$GD(3, 4)$
52	10	25	10	4	3	6	8	1	6	8	361	40	$GD^*(5, 2)$
53	10	25	10	4	4	2	6	3	3	4	417	25	$T(5)$
54	8	16	10	5	6	4	6	1	4	6	92	95	$GD^*(4, 2)$
55†	12	20	10	6	4	5	5	6	4	0	22	100	$GD(2, 6)$
56	8	20	10	4	3	6	4	3	0	4	14	85	$GD^*(2, 4)$
57	8	20	10	4	4	6	6	1	4	6	12	85	$GD^*(4, 2)$
58	10	20	10	5	4	5	5	4	0	5	135	100	$GD^*(2, 5)$
59	18	20	10	9	4	5	5	12	4	0	718	95	$GD(3, 6)$
60†	9	33	11	3	2	3	2	6	1	0	63	100	$GD(3, 3)$
61†	9	33	11	3	2	5	6	2	3	6	272	100	$GD^*(3, 3)$
62	8	22	11	4	5	3	6	1	4	6	9	100	$GD^*(4, 2)$
63	10	22	11	5	4	5	1	8	0	0	12	100	$GD(5, 2)$
64	14	22	11	7	5	6	12	1	10	12	654	55	$GD^*(7, 2)$
65†	12	48	12	3	2	3	9	2	6	9	510	55	$GD^*(4, 3)$
66†	12	48	12	3	2	4	10	1	8	10	720	30	$GD^*(6, 2)$
67	10	40	12	3	2	4	6	3	3	4	77	5	$T(5)$
68	10	40	12	3	3	0	8	1	6	8	71	10	$GD^*(5, 2)$
69	8	32	12	3	3	6	6	1	4	6	9	80	$GD^*(4, 2)$
70†	12	24	12	6	5	6	6	5	0	6	21	100	$GD^*(2, 6)$
71†	12	24	12	6	6	4	8	3	4	8	421	70	$GD^*(3, 4)$
72	10	20	12	6	6	8	6	3	3	4	472	10	$T(5)$
73	10	20	12	6	7	4	8	1	6	8	253	60	$GD^*(5, 2)$
74	9	18	12	6	8	6	6	2	3	6	358	15	$GD^*(3, 3)$
75	8	24	12	4	5	6	6	1	4	6	18	100	$GD^*(4, 2)$
76	10	24	12	5	5	6	6	3	3	4	13	100	$T(5)$
77†	9	39	13	3	3	4	6	2	3	6	172	100	$GD^*(3, 3)$
78†	9	39	13	3	4	1	6	2	3	6	162	10	$GD^*(3, 3)$
79†	8	26	13	4	5	6	3	4	2	0	9	100	$GD(2, 4)$
80	10	26	13	5	6	4	8	1	6	8	149	80	$GD^*(5, 2)$
81†	9	39	13	3	4	1	6	2	3	6	162	10	$GD^*(3, 3)$
82†	8	26	13	4	5	6	3	4	2	0	9	100	$GD(2, 4)$

Note. Designs marked † are resolvable

Table 1. PBIBD(2)s obtained by TS – continued

No.	v	b	r	k	$\lambda_1$	$\lambda_2$	$n_1$	$n_2$	$p_{11}^1$	$p_{11}^2$	ITO	POS	AS
83	10	26	13	5	6	4	8	1	6	8	149	80	$GD^*(5, 2)$
84	12	26	13	6	5	6	1	10	0	0	29	100	$GD(6, 2)$
85†	9	42	14	3	3	5	6	2	3	6	442	85	$GD^*(3, 3)$
86†	9	42	14	3	4	2	6	2	3	6	435	15	$GD^*(3, 3)$
87	10	35	14	4	4	6	6	3	3	4	80	5	$T(5)$
88	10	20	14	7	10	8	6	3	3	4	186	45	$T(5)$
89	10	28	14	5	6	8	8	1	6	8	157	90	$GD^*(5, 2)$
90	12	28	14	6	5	8	6	5	0	6	465	40	$GD^*(2, 6)$
91	14	28	14	7	6	7	7	6	0	7	1334	40	$GD^*(2, 7)$
92†	9	45	15	3	3	4	2	6	1	0	252	100	$GD(3, 3)$
93	8	40	15	3	3	6	4	3	0	4	61	25	$GD^*(2, 4)$
94	10	50	15	3	3	6	8	1	6	8	12	68	$GD^*(5, 2)$
95	10	50	15	3	4	2	6	3	3	4	12	10	$T(5)$
96	8	40	15	3	4	6	6	1	4	6	8	100	$GD^*(4, 2)$
97†	8	30	15	4	6	7	4	3	0	4	10	100	$GD^*(2, 4)$
98†	8	30	15	4	7	3	6	1	4	6	78	95	$GD^*(4, 2)$
99	14	35	15	6	6	3	12	1	10	12	1095	40	$GD^*(7, 2)$
100	10	25	15	6	9	7	6	3	3	4	507	30	$T(5)$
101	12	45	15	4	4	5	10	1	8	10	151	95	$GD^*(6, 2)$
102	10	30	15	5	6	7	3	6	0	1	17	100	$T^*(5)$
103	12	30	15	6	6	7	2	9	1	0	79	100	$GD(4, 3)$
104	14	30	15	7	6	7	1	12	0	0	215	100	$GD(7, 2)$
105	10	40	16	4	4	7	5	4	0	5	332	55	$GD^*(2, 5)$
106	10	40	16	4	5	8	8	1	6	8	18	45	$GD^*(5, 2)$
107	10	40	16	4	6	4	6	3	3	4	32	10	$T(5)$
108†	16	32	16	8	7	8	8	7	0	8	79	100	$GD^*(2, 8)$
109	8	32	16	4	6	7	1	6	0	0	21	100	$GD(4, 2)$
110	10	32	16	5	7	8	8	1	6	8	25	100	$GD^*(5, 2)$
111	12	32	16	6	7	8	8	3	4	8	68	100	$GD^*(3, 4)$
112†	9	51	17	3	4	5	6	2	3	6	259	100	$GD^*(3, 3)$
113†	10	34	17	5	7	8	4	5	3	0	41	100	$GD(2, 5)$
114	12	34	17	6	7	8	3	8	2	0	74	100	$GD(3, 4)$
115	16	34	17	8	7	8	1	14	0	0	579	15	$GD(8, 2)$
116	12	36	18	6	8	9	9	2	6	9	139	100	$GD^*(4, 3)$
117†	10	38	19	5	8	9	5	4	0	5	17	100	$GD^*(2, 5)$
118	8	38	19	4	8	9	6	1	4	6	25	100	$GD^*(4, 2)$
119	12	38	19	6	9	5	10	1	8	10	743	55	$GD^*(6, 2)$
120†	8	40	20	4	8	9	3	4	2	0	13	100	$GD(2, 4)$
121	10	40	20	5	8	9	1	8	0	0	13	100	$GD(5, 2)$
122	12	40	20	6	9	10	10	1	8	10	85	95	$GD^*(6, 2)$

Note. Designs marked † are resolvable