Neural Cost Functions and Search Strategies for the Generation of Block Designs: an Experimental Evaluation

Pau Bofill

Departament d'Arquitectura de Computadors (UPC), Campus Nord- Mòdul-D6, c/ Gran Capità s/n, 08071 Barcelona, pau@ac.upc.es

Carme Torras

Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Edifici Nexus, c/ Gran Capità 2-4, 08034 Barcelona, ctorras@iri.upc.es

Abstract

A constraint satisfaction problem, namely the generation of Balanced Incomplete Block Designs (v, b, r, k, λ)-BIBDs, is casted in terms of function optimization. A family of cost functions that both suit the problem and admit a neural implementation is defined. An experimental comparison spanning this repertoire of cost functions and three neural relaxation strategies (Down-Hill search, Simulated Annealing and a new Parallel Mean Search procedure), as applied to all BIBDs of up to 1000 entries, has been undertaken. The experiments were performed on a Connection Machine CM-200 and their analysis required a careful study of performace measures. The simplest cost function standed out as the best one for the three strategies. Parallel Mean Search, with several processors searching cooperatively in parallel, could solve a larger number of problems than the same number of processors working independently, but Simulated Annealing yielded overall the best results. Other conclusions, as detailed in the paper, could be drawn from the comparison, BIBDs remaining a challenging problem for neural optimization algorithms.

Keywords: block designs, neural cost functions, simulated annealing, parallel mean search, performance measures, experimental evaluation

1 Introduction

The work described in this paper originates in [6], where the generation of block designs was used as a benchmark for comparing the performance of several related optimizing search strategies, based on neural networks. Such

strategies were approached as general purpose techniques. Therefore, no attempt was made to build any problem knowledge (i.e, design properties) into the search strategies. This paper summarizes the results obtained.

1.1 The problem: Block Designs

Balanced Incomplete Block Designs (BIBDs) have their origins in the field of Experimental Design, and their properties and generation are studied by Combinatorial Analysis [12,28]. Taking (v, b, u) as independent parameters, and in terms of its incidence matrix, a *Balanced Incomplete Block Design* (v, b, u)-BIBD can be defined as follows. Let $A \equiv [x_{ij}]$ be a given configuration in the space $\mathcal{A} - v \times b$ of binary configurations with v rows and b columns. Let $x_{ij} \in \{0, 1\}$, the state variables, represent the incidence of treatment i in block j of A, and let $o = \sum_{i=1}^{v} \sum_{j=1}^{b} x_{ij}$ be the number of ones in A (the number of plots), $r_i = \sum_{j=1}^{b} x_{ij}$ the number of ones in row i (the replicate number for treatment i), $k_j = \sum_{i=1}^{v} x_{ij}$ the number of ones in column j (the size of block j), and $\lambda_{il} = \sum_{j=1}^{b} x_{ij} x_{lj}$ the correlation or dot product between rows i and l (the number of times that treatments i and l occur together in a block).

Definition 1 *For fixed r, k and λ, with $k < v$ and $\lambda > 0$, we say that A is the incidence matrix of a BIBD with parameters (v, b, u) and descriptors $[r, k, \lambda]$* if and only if *the following properties are fulfilled:*

i) Right number of ones: $o = u$.
ii) Strictly uniform rows: $r_i = r, \quad i = 1, \ldots, v$.
iii) Strictly uniform columns: $k_j = k, \quad j = 1, \ldots, b$.
iv) Strict balance: $\lambda_{il} = \lambda, \quad i = 1, \ldots, v-1, l = i+1, \ldots, v$.

Parameters and descriptors are related by the following *multiplicity* conditions,

$$
\begin{aligned}
r &= u/v \\
k &= u/b \\
\lambda &= \frac{r(k-1)}{v-1} = \frac{u(u-b)}{bv(v-1)},
\end{aligned}
$$

with r, k and λ *integers*, therefore restricting the range of *admissible* parameter sets.

The admissibility of its parameters is a necessary but not sufficient condition for the *existence* of a block design. The situation is summarized in [18], that lists all non-trivial admissible parameter sets with $r \leq 41$, together with the currently known bounds on the number N_s of *non-isomorphic* solutions. In particular, whenever it has been established that a particular design does not exist $N_s = 0$, and $N_s =?$ denotes an unsettled case. For our use here all

cases with $vb \leq 1000$ are listed in Appendix A. Some (infinite) families of block designs (designs whose parameters satisfy particular properties) can be constructed analytically, by direct or recursive methods [12, Chapter 15], and the state of the art in computational methods for design generation is described in [9]. The smallest unsettled case is $(22, 33, 264)$ [20], with $vb = 726$ entries, showing that exhaustive search is still intractable for designs of this size. In the general case, as with other combinatorial configurations, the algorithmic generation of block designs is an NP problem [8].

Several alternative combinatorial configurations have been defined in the literature for the experimental settings where the desired parameters are not admissible (i.e, (v, b, u) leading to non-integer $[r,\ k,\ \lambda]$), the most usual being *Pairwise Balanced Designs* [24], (r, λ)-designs [17], *Partially Balanced Incomplete Block Designs* [28,29] and *Regular Graph Designs* [16]. In [6,5] a new family of similar combinatorial configurations, *Maximally Balanced Maximally Uniform Designs* (v, b, u)-MBMUDs, arises as the natural generalization of BIBDs implied by our study of neural cost functions. MBMUDs allow for at most two consecutive values for its row, column and balance descriptors $[r,\ r+1;\ k,\ k+1;$ and $\lambda,\ \lambda+1]$.

1.2 The tool: Optimizing Neural Networks

The generation of block designs is a constraint satisfaction problem. In order to use optimizing neural networks we must first reformulate it as a combinatorial optimization problem and then map it onto a standard neural network architecture. The set of cost functions that are described in this work are based on the number of *pairs* (the number of active *connections*) as a distribution measure for each of the properties of a BIBD [5], therefore mapping straightforwardly onto an optimizing network which, because of the balance property, uses connections of arity four.

Three search strategies are considered. Two standard ones, Down-Hill search DH [14] and Simulated Annealing SA [3], plus a novel strategy, Parallel Mean Search PMS [4], which is described in Section 3.2.

The application of neural networks to combinatorial optimization problems (Hopfield networks) was first proposed in [15] and is deeply analyzed in [1]. Networks with higher arity connections are considered in [27]. Simulated Annealing (or Boltzmann machines) and its deterministic version, Mean Field Annealing MFA [25], are based on statistical mechanics models and, therefore, they have a strong theoretical background [2,13]. Some variants and applications of the MFA model can be found in [26,11]. The dynamics of optimizing neural networks are studied in [10], among others. A most general framework

for Hopfield networks is defined in [30].

1.3 Paper Overview

In Section 2 a family of cost functions for BIBD generation is defined. Section 3 describes the three search strategies, with special emphasis on the new Parallel Mean Search procedure. Section 4 describes the experimental setting for algorithm comparison, together with the results. Finally, Section 5 is devoted to conclusions.

2 Neural Cost Functions for the Generation of Block Designs, based on Distribution Measures

Let $\mathcal{A} - v \times b$ be the set of all $A \equiv [x_{ij}]$ binary *configurations* with v rows and b columns. We say that $F : \mathcal{A} \to \mathcal{R}$ is a *cost function* for the generation of block designs if there exists a *lower bound* F^* such that $F(A^*) = F^*$ *if and only if* there exists a (v, b, u)-BD with incidence matrix A^*.

The cost functions considered in this work [5] are defined as the linear combination of a set of distribution measures for each of the properties of a block design.

$$F(A) = \rho_u U(A) + \rho_t P_t(A) + \rho_h P_h(A) + \rho_v P_v(A) + \rho_q Q(A) + \rho_{\bar{q}} \bar{Q}(A), \quad (1)$$

with U, P_t, P_h, P_v, Q and $\bar{Q}$ defined next, and $(\rho_u, \rho_t, \rho_h, \rho_v, \rho_q, \rho_{\bar{q}})$, the coefficients of the linear combination, defining the *composition* of a particular cost function.

$$\textit{Number of ones:}\ U(A) = o$$

$$\textit{Total pairs of ones:}\ P_t(A) = \binom{o}{2}$$

$$\textit{Horizontal pairs of ones:}\ P_h(A) = \sum_i \binom{r_i}{2}$$

$$\textit{Vertical pairs of ones:}\ P_v(A) = \sum_j \binom{k_j}{2}$$

$$\textit{Quadruples of ones:}\ Q(A) = \sum_{i=1}^{v-1} \sum_{l=i+1}^{v} \binom{\lambda_{il}}{2}$$

$$\textit{Quadruples of zeros: } \bar{Q}(A) = \sum_{i=1}^{v-1} \sum_{l=i+1}^{v} \binom{\bar{\lambda}_{il}}{2}$$

with $\bar{Q}(A) = Q(\bar{A})$, $\bar{A} \equiv [\bar{x}_{ij}]$ the bit-wise *complementary* configuration of A, and $\bar{\lambda}_{il} = b + \lambda_{il} - r_i - r_l$, the correlation between rows i and l in $\bar{A}$.

The measures U and P_t are a linear and a quadratic function, respectively, of the number o of ones in A. Since P_h is quadratic on the r_i's, it is a measure of the distribution of ones over rows. For fixed o, P_h is minimum when rows are *maximally* or *uniformly* distributed, that is, when the r_i's are as even as possible. In this case, for arbitrary o, the r_i's take at most two consecutive values ($\lfloor o/v \rfloor$ and $\lfloor o/v \rfloor + 1$), and when $o|v$ (multiplicity) all rows have exactly $\lfloor o/v \rfloor$ ones. Likewise, minimizing P_v for a fixed number of ones leads to *maximally uniform* columns. Measure Q takes into account the distribution of vertical pairs of ones over pairs of rows and, for a fixed o, it is minimum when columns are maximally uniform *and* correlations of ones are *maximally balanced*. That is, when the λ_{il}'s take at most two distinct consecutive values. Finally, $\bar{Q}$ measures balance in $\bar{A}$.

The *local increments* of a generic measure M are defined as follows. We say that configurations A and A' are *neighbours* if they differ in *only one* component, namely component (i, j). Let x_{ij} denote its value in A, and let $\bar{x}_{ij} = 1 - x_{ij}$ be its value in A'. Then,

$$\Delta^{ij} M(A) = M|_{x_{ij}=1} - M|_{x_{ij}=0} = \begin{cases} M(A) - M(A'), & \text{when } x_{ij} = 1 \\ M(A') - M(A), & \text{when } x_{ij} = 0. \end{cases}$$

Therefore, local increments of measure F are defined as

$$\begin{aligned} \Delta^{ij} F(A) = {} & \rho_u \Delta^{ij} U(A) + \rho_t \Delta^{ij} P_t(A) + \rho_h \Delta^{ij} P_h(A) + \\ & + \rho_v \Delta^{ij} P_v(A) + \rho_q \Delta^{ij} Q(A) + \rho_{\bar{q}} \Delta^{ij} \bar{Q}(A). \end{aligned} \tag{2}$$

For further reference, local increments are related to *transition* increments by

$$\Delta^{ij}_{\text{trans}} M(A) = (1 - 2x_{ij}) \Delta^{ij} M(A),$$

with

$$\Delta^{ij}_{\text{trans}} M(A) = M|_{\bar{x}_{ij}} - M|_{x_{ij}} = M(A') - M(A).$$

When the parameter set (v, b, u) is *admissible* the *optimal values* of the previous measures are derived by assuming that properties i) to iv) in definition 1 hold (*optimality assumption*), even if the corresponding design does not exist. Furthermore, under the same assumption and for all measures above,

Table 1
Given a set (v, b, u) of admissible parameters, optimal values and local increments for U, P_t, P_h, P_v, Q and $\bar{Q}$, expressed in terms of the descriptors of the design (with $\bar{k} = v - k$ and $\bar{\lambda} XS = b + \lambda - 2r$).

M	M^*	$\Delta^1 M^*$	$\Delta^0 M^*$
U	u	1	1
P_t	$\binom{u}{2}$	$u-1$	u
P_h	$v\binom{r}{2}$	$r-1$	r
P_v	$b\binom{k}{2}$	$k-1$	k
Q	$\binom{v}{2}\binom{\lambda}{2}$	$(k-1)(\lambda-1)$	$k\lambda$
$\bar{Q}$	$\binom{v}{2}\binom{\bar{\lambda}}{2}$	$-\bar{k}\bar{\lambda}$	$-(\bar{k}-1)(\bar{\lambda}-1)$

optimal local increments can be shown to take only two distinct values, depending only on the *state* (0 or 1) of the corresponding component [5]. These optimal local increments are generically denoted $\Delta^1 M^* \equiv \Delta^{ij} M^*|_{x_{ij}=1}$ and $\Delta^0 M^* \equiv \Delta^{ij} M^*|_{x_{ij}=0}$, and, together with optimal values, they are listed in Table 1.

The composition coefficients of measure F in (1) must satisfy some constraints in order to get a valid cost function:

Theorem 2 *Given a set (v, b, u) of admissible parameters, the measure F in equation (1) is a cost function for the generation of BIBDs* if

$$\begin{aligned} \rho_q &> 0 \\ \rho_t, \rho_h, \rho_v, \rho_{\bar{q}} &\geq 0 \end{aligned}$$

and

$$\begin{aligned} &\Delta^1 F^* < 0 \\ &\Delta^0 F^* > 0. \end{aligned}$$

Its global minimum is

$$F^* = \rho_u U^* + \rho_t P_t^* + \rho_h P_h^* + \rho_v P_v^* + \rho_q Q^* + \rho_{\bar{q}} \bar{Q}^*, \tag{3}$$

with the optimal values in Table 1.

The proof can be found in [5].

We say that a cost function is *symmetric* when

$$\Delta^1 F^* = -\Delta^0 F^*.$$

In order to restrict the number of possibilities, only symmetric cost functions were considered in the experimental comparison.

Of particular interest is cost function $F_{uq} \equiv F(\alpha_u, 0, 0, 0, 1, 0)$, with

$$\alpha_u = \frac{\Delta^1 Q^* + \Delta^0 Q^*}{2},$$

the symmetric case setting for ρ_u. This function includes the minimum subset of distribution measures strictly required by theorem 2 (i.e, the U and Q terms) and, therefore, it represents the *core* or simplest cost function for BIBD generation [5].

The above cost functions are isomorphic to an *optimizing neural network* if a neural unit is defined for each x_{ij} state variable, and all connections (explicit interactions) of the same type are weighted by the corresponding coefficient in F. Thus, function F corresponds to the energy E of the network and local increments $\Delta^{ij} F$ correspond to local fields ϕ_{ij}. Notice that quadruples describe arity four interactions, leading to higher (4th) order networks. In the following, the terms energy and function cost will be used indistinctly.

3 Search Strategies

Given a set (v, b, u) of admissible parameters, the search space $\mathcal{A} - v \times b$ has 2^{vb} possible configurations. The x_{ij} variables represent the *current state* of the search, and the weighted sum of the connections evaluates the current energy $E(A)$ of configuration A. Connection weights codify the composition coefficients of the cost function chosen. Therefore, optimal energy values, current energy values and local fields are given by equations (3), (1), and (2), respectively.

For optimizing networks, the basic exploration principle is *local search*, which allows transitions only between neighbour states. In this framework, a *search strategy* is a set of criteria for selecting an initial state, a unit updating order, a decision rule (either to accept or to refuse transitions), and a stopping condition. And the result of the search is either failure or success. Depending on the strategy, some of these criteria are parametric. In this work, the initial state will always be selected at random, and units will be updated in a fixed

sequential order (by rows), thus confining all randomness in the selection of the initial state. The remaining operations depend on each particular strategy.

We say that a *descent* is a whole run of the search algorithm. An *iteration* is one update of each unit. An *update* is the evaluation of the local field and the application of the decision rule. And, whenever the decision rule accepts it, a *transition* is the commutation of a unit and the corresponding energy update.

3.1 Down-Hill Search (DH) and Simulated Annealing (SA)

The basic idea of Down-Hill search is to accept all energy-decreasing transitions, until an optimum is found or the algorithm converges to a minimum. The decision rule

$$x_{ij} \leftarrow \bar{x}_{ij} \quad \text{iff } \Delta^{ij}_{\text{trans}} E < 0,$$

depends only on the *sign* of $\Delta^{ij}_{\text{trans}} E$. Down-Hill search is the *basic* exploration strategy upon which the other strategies are constructed, and it is not parametric.

The goal of Simulated Annealing is to avoid undesired local minima by means of *thermal noise*. If the Metropolis decision rule is used [22], the *probability* of accepting a transition at *computational temperature* T is given by

$$P\{x_{ij} \rightarrow \bar{x}_{ij}\} = \begin{cases} 1, & \Delta^{ij}_{\text{trans}} E < 0 \\ e^{-\frac{\Delta^{ij}_{\text{trans}} E}{T}}, & \text{otherwise.} \end{cases}$$

The efficacy of Simulated Annealing depends on a good temperature *schedule*. In practice (see, for instance [11]), results are good with a *smooth* decrement law such as

$$T_k = \sqrt[vb]{\tau} T_{k-1},$$

where k represents the number of updated *units*, and τ is the decay constant corresponding to a complete iteration. The underlying assumption is that, in that way, perturbations on thermal equilibrium will be small.

If no solution is found before, the stochastic phase is stopped after N_t iterations and the algorithm continues with Down-Hill search, until a minimum is found. Thus, control variables for SA are N, the current number of iterations, and T, the current temperature. In terms of the initial T_0 and final T_f temperatures, the strategy parameters are defined to be the following: the maximum number of iterations of the stochastic phase $N_t = \frac{\ln T_f/T_0}{\ln \tau}$, the *central* temperature

$T_c = \sqrt{T_0 T_f}$, and the temperature *range* $\rho = \frac{T_0}{T_f}$. Such a parameter formulation allows us to define *a priori* the desired invested effort on the stochastic phase, and it should also be helpful in experimentally finding a T_c value near to the critical temperature T_{cri} where global minima are formed. A particular instance of Simulated Annealing is then denoted SA(T_c,ρ,N_t).

3.2 The New Strategy: Parallel Mean Search (PMS)

Parallel Mean Search [4,6], like Simulated Annealing, is based on the assumption that global optima are located in regions of low average energy. Instead of thermal exploration, though, Parallel Mean Search looks for these regions by *sampling* the search space in parallel, using several instances of the network moving together as a *cluster* in the direction of decreasing average energy, like a big sliding ball. The gradual reduction of the radius (maximum Hamming distance to the center of the cluster), leads the search into deeper and deeper mean energy basins. Parallel Mean Search (also called *Cooperative Search* in [6]) is especially suitable for SIMD architectures because, having a reasonably low communication cost, it effectively exploits the *cooperation* between the cluster members.

The *energy of the cluster* $\mathcal{E}$ is defined as the sum of the energies of its members

$$\mathcal{E} = \sum_{p=1}^{S} E^p,$$

with S the size of the cluster and E^p the energy of member p, thus simultaneously sampling S points in search space. At each unit update, the next unit (i,j) is selected (the *same* unit for all members), and a transition is accepted for *each* of them if the *cluster's* energy decreases. If the transition energy of the cluster with respect to (i,j) is defined as

$$\Delta^{ij}_{\text{trans}}\mathcal{E} = \sum_{p=1}^{S} \Delta^{ij}_{\text{trans}} E^p,$$

then a transition is accepted or not according to the decision rule

$$x^p_{ij} \leftarrow \bar{x}^p_{ij} \quad \text{iff } \Delta^{ij}_{\text{trans}}\mathcal{E} < 0, \qquad \text{for } p = 1, \ldots, S.$$

Thus, the cluster as a whole performs Down-Hill search and, at every transition, although the energy of some of its members may increase, the overall cluster energy decreases. Since all members update exactly the same unit (they all move in the same direction), the *topology* of the cluster remains unchanged.

The size of the sampled region is governed by the radius R of the cluster, de-

fined as the maximum Hamming distance between any of its members and the *center* of the cluster. At each radius decrement, the cluster *contracts* towards the center. The efficacy of the search, then, will depend on a good choice of the initial R_0 and final R_f radii, and on a good reduction *schedule*. If no optimum has been found before, the maximum number of iterations of the *cooperative phase* is fixed to N_t. After that, the cluster is released and each member relaxes independently using Down-Hill search until it reaches a local minimum. The cluster is defined to *succeed* if *any of its members finds a solution*. The choice of the size S of the cluster will also be relevant. In this work we have chosen to keep it fixed along the search process. Thus, the sampling density increases as R is reduced.

The topology of the cluster and the contraction mechanism offer several alternatives. The first topology considered here consists of initializing all cluster members to the same initial random state (the *center* of the cluster), selecting R units at random (the *same* R units for all members), and setting them at a random value, independently for each cluster member. In this way, all members share the value of $vb - R$ units, and with the remaining R (the *variable* units), the cluster uniformly samples a subspace of cardinality 2^R. We call this a *focused* topology, since all members focus on the same subspace, and the resulting strategy will be denoted PMSf.

The other proposed choice, which will be called *spread* topology (denoted PMSs), is constructed in the same way except that the R variable units are selected *independently* for each cluster member. Thus, there is not a fixed set of common units and, around the central point, the members spread uniformly in any of the vb dimensions, sampling the search space within a distance R from the center.

When the radius is reduced, the *contraction* mechanism consists of selecting one of the variable units in each member, and set it to a common value. For the focused clusters, the selected unit (i, j) is the same for all members, chosen at random out of R. But the value it is assigned can be selected in two ways. The first, which we call *least energy* contraction, consists of choosing the value (0 or 1) that minimizes the energy of the cluster, according to

$$x_{ij}^p \leftarrow \begin{cases} 1 & \text{if } \Delta^{ij}\mathcal{E} < 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } p = 1, \ldots, S,$$

with

$$\Delta^{ij}\mathcal{E} = \mathcal{E}|_{x_{ij}^p=1} - \mathcal{E}|_{x_{ij}^p=0} = \sum_{p=1}^{S} \Delta^{ij} E^p,$$

the *local energy increment* of the cluster with respect to (i, j). Notice that,

although the least energy setting is selected for x_{ij}, the cluster as a whole may increase its energy. This variant will be denoted PMSfl.

The second choice, which we call *central* contraction (denoted PMSfc), consists of actually keeping a *central member* q, which acts as a reference, and copying its value to the remaining members. That is, with (i, j) the selected unit,

$$x_{ij}^p \leftarrow x_{ij}^q, \quad \text{for } p = 1, \ldots, S.$$

In both cases, the energy of the cluster must be updated taking into account the members that have actually changed their values. If y denotes the new assigned value, then

$$\mathcal{E} \leftarrow \mathcal{E} + \sum_{p=1}^{S} (x_{ij}^p \oplus y) \Delta_{\text{trans}}^{ij} E^p,$$

with $(x_{ij}^p \oplus y)$ the exclusive-or between each of the old values, and the new common value.

For spread clusters, the unit selected for contraction is different for each member p, and it is chosen at random out of its own R variable units. Let (i_p, j_p) denote the selected unit at p. Least energy contraction then is no longer possible and, with q the reference member, central contraction is performed according to

$$x_{i_p j_p}^p \leftarrow x_{i_p j_p}^q \quad \text{for } p = 1, \ldots, S,$$

with energy updated as

$$\mathcal{E} \leftarrow \mathcal{E} + \sum_{p=1}^{S} (x_{i_p j_p}^p \oplus x_{i_p j_p}^q) \Delta_{\text{trans}}^{i_p j_p} E^p.$$

For the radius decrement schedule, a linear law with smooth temporal granularity has been selected. Since R must be an integer, we use an auxiliary continuous control variable $r \in \mathcal{R}$, and we update the radius according to its integer part. Thus,

$$r_k = r_{k-1} - \frac{\Delta r}{vb},$$

with k the number of updated *units*, and Δr the step corresponding to a complete iteration, and we take $R_k = \lfloor r_k \rfloor$. After each update, the cluster must contract $R_{k-1} - R_k$ times, which may be none or several. Like before, given the initial R_0 and the final R_f radii, the step is actually determined by the maximum number of allowed iterations of the cooperative phase,

$$N_t = \frac{R_0 - R_f}{\Delta r}.$$

Parallel Mean Search is thus presented in three variants PMSfl, PMSfc and PMSs, and it is parametric on PMS(S, R_0, R_f, N_t). Its control variables are N, r and R. In the case of Parallel Mean Search, the local search basic procedure is executed by each cluster member in parallel, and cluster energy operations involve some (although little) communication.

4 Experiments and Results

This section describes the experimental comparison of the proposed search *algorithms* (functions plus strategies) as applied to the *problem* of block design generation. In terms of experimental analysis, the *experimental search space* is defined by the three main experimental *factors*: problem, function and strategy. Each particular set of admissible design parameters constitutes a level of the problem factor, yielding infinitely many problem instances. Function levels are defined by the composition coefficients (again, infinite choices). Finally, strategies are organized as subfactors (namely DH, SA, and the three variants of PMS), with their corresponding levels defined by their (infinitely many) parameter settings.

Considering all this, an exhaustive analysis is obviously intractable, and the experimentation is planned in *three stages*: A training stage, for function selection and parameter tuning. A comparison stage between the proposed strategies, and a third stage, where the best performing algorithm in the previous stage is applied to problems of increasing size.

4.1 Definition of the Response Variable of an Experiment

The *expected number of runs to the first solution* is a measure of the *efficacy* of a search algorithm (or, reciprocally, of the *difficulty* of a problem). But it does not take into account the resources invested by different algorithms. Thus, for an objective comparison, the *expected cost to the first solution* must be used, as a measure of the *efficiency* of the search. In this work, since all three strategies are based in local search, the computational complexity of an iteration is the same for all of them (the quadruple term of the local increments dominates, yielding $\mathcal{O}(v^2b^2)$ [6]). Thus, in order to avoid implementation issues, computational cost can be compared in terms of the number of invested *iterations*. Deciding the outcome of the search needs no further computation, since BIBDs are identified by their optimal energy value (notice that, since isomorphism is not considered, any solution is as good as another).

For a given experimental case (a particular problem, function and strategy),

the *elementary experiment* is defined as a single run or *descent* of the algorithm, with *outcome* $\mathbf{x} \in \Omega = \{0, 1\} = \{\text{failure, success}\}$, and *computational cost* $\mathbf{c}$ (in iterations). Since (at least) the choice of the initial state is random, $\mathbf{x}$ is a Bernoully variable, with success probability p and failure probability $q = 1 - p$, and $\mathbf{c}$ is a random variable of unknown distribution, with expected value $E(\mathbf{c})$ and variance $Var(\mathbf{c})$.

Next, we define a *sequential experiment* as the replication of the elementary experiment until the first solution is found. Since replications are independent from each other,

$$\mathbf{y} \equiv \{ \text{ Number of descents to the first solution } \}$$

is a geometric random variable with parameter p and

$$\begin{aligned} E(\mathbf{y}) &= 1/p \\ Var(\mathbf{y}) &= q/p^2. \end{aligned}$$

Thus, $\mathbf{y}$ is a direct measure of the *efficacy* of the search.

In order to take cost into account, we define

$$\mathbf{z} \equiv \{ \text{ Computational cost to the first solution } \},$$

which, with $\mathbf{c}_j$ the cost of descent j, can be expressed as

$$\mathbf{z} = \sum_{j=1}^{\mathbf{y}} \mathbf{c}_j.$$

This variable, thus, is a direct measure of the *efficiency* of the search.

The actual cost c of a descent (as was verified experimentally) depends on the corresponding outcome x. So we define $\mathbf{b} \equiv \mathbf{c}|_{x=1}$ as the cost of the successful descents, and $\mathbf{d} \equiv \mathbf{c}|_{x=0}$ the cost of the unsuccessful ones. Although the $\mathbf{c}_j$'s above are independent from each other, they are *not* independent from $\mathbf{y}$ (only the *last* descent is successful). Then, under these assumptions, the expected value and variance of $\mathbf{z}$ can be written as

$$\begin{aligned} E(\mathbf{z}) &= (E(\mathbf{y}) - 1)E(\mathbf{d}) + E(\mathbf{b}) \\ Var(\mathbf{z}) &= Var(\mathbf{y})E^2(\mathbf{d}) + (E(\mathbf{y}) - 1)Var(\mathbf{d}) + Var(\mathbf{b}). \end{aligned}$$

Finally, we define a *parallel experiment* as the simultaneous execution of N elementary experiments. Since the N descents of the algorithm are now performed at the same time, although the outcome and cost of each elementary experiment are available, it is not possible to tell *how many* of the unsuccessful outcomes are associated with each of the successful ones. In other words,

the random variables $\mathbf{y}$ and $\mathbf{z}$ are not directly observable. In this context, the random variable

$$\mathbf{X} \equiv \{ \text{ Number of successes in } N \text{ descents } \},$$

which may be expressed as

$$\mathbf{X} = \sum_{i=1}^{N} \mathbf{x}_i,$$

is binomial with

$$\begin{aligned} E(\mathbf{X}) &= Np \\ Var(\mathbf{X}) &= Npq. \end{aligned}$$

The best estimator of the success probability p is, therefore,

$$\hat{\mathbf{p}} = \frac{\mathbf{X}}{N},$$

with expected value $E(\hat{\mathbf{p}}) = p$ and variance $Var(\hat{\mathbf{p}}) = pq/N$. The statistics of $\mathbf{y}$ can then be *estimated* by the estimates of its parameters as

$$\begin{aligned} \hat{E}(\mathbf{y}) &= 1/\hat{p} \\ \widehat{Var}(\mathbf{y}) &= \hat{q}/\hat{p}^2, \end{aligned}$$

with $\hat{q} = 1 - \hat{p}$, and the statistics of $\mathbf{z}$ are estimated by

$$\begin{aligned} \hat{E}(\mathbf{z}) &= (\frac{1}{\hat{p}} - 1)(\overline{d}) + (\overline{b}) \\ \widehat{Var}(\mathbf{z}) &= \frac{\hat{q}}{\hat{p}^2}(\overline{d})^2 + (\frac{1}{\hat{p}} - 1)(s^2_{\mathbf{d}}) + (s^2_{\mathbf{b}}), \end{aligned} \tag{4}$$

with $\overline{b}$ and $S^2_{\mathbf{b}}$ the sample mean and variance of the X successful descents, and $\overline{d}$ and $S^2_{\mathbf{d}}$ the sample mean and variance of the $N - X$ unsuccessful ones.

With $\mathbf{C} = \sum_{i=1}^{N} \mathbf{c}_i$ the total cost of the parallel experiment, equation (4) is equivalent to $\hat{E}(\mathbf{z}) = \frac{C}{X}$. Then, the random variable

$$\mathbf{w} = \frac{\mathbf{C}}{\mathbf{X}}$$

verifies that, for $N \to \infty$,

$$E(\mathbf{w}) \to E(\mathbf{z}),$$

and it is therefore an asymptotically consistent *estimator* of $E(\mathbf{z})$. Since $E(\mathbf{z})$ is the desired measure of efficiency, we define $\mathbf{w}$ as the *response* variable of

the parallel experiment. Its variance $Var(\mathbf{w})$ determines the precision of the estimate, and it will be evaluated experimentally.

A pathological case of the parallel experiment occurs when *none* of the N descents is succesful (i.e, when $X = 0$). This circumstance is termed hereafter a *nil* result and it prevents the computation of $\mathbf{w}$.

All experiments in this work have been performed on a Connection Machine CM-200 [7], with 2048 bit serial processors (which fit well the binary state variables), and simulations have been arranged so that each processor performs a single descent of the algorithm. Thus, in all, each execution corresponds to a parallel experiment with $N = 2048$ descents.

The above experimental measures also apply to Parallel Mean Search if the elementary experiment is defined as a descent of the whole *cluster* (S non-independent member descents). The descent is defined to be successful if *any* of the cluster members finds a solution, and the cost of the descent is the *sum* of the costs of the cluster members. In that way, the expected cost to the first solution effectively takes into account all the invested resources. For comparison purposes, parallel experiments with PMS where performed with $N = 2048$ independent *clusters*.

4.2 First Stage: Training

In a set of preliminary experiments [6], the response variable $\mathbf{w}$ was found to be an accurate enough estimator of $E(\mathbf{z})$, but it had the following drawbacks: it sometimes had a non-normal distribution, it had different variances on different cases and, as mentioned, it could not quantitatively deal with *nil* results. Therefore, objective comparison methods such as ANOVA [23] could not be used. In the following, several replications of the *parallel* experiment were taken for each experimental case in order to measure the *mean* and *deviation* of $\mathbf{w}$ in each case, and comparisons were made in terms of the means. For a given comparison, whenever the number of *nil* results was equal, a difference in means was considered to be "*significant*" when its absolute value was larger than the *sum* of the corresponding deviations. Otherwise, the setting with less *nil* results was considered to perform better.

The goal of the *training stage* was to reduce the size of the experimental search space. A *test problem set* was chosen with the 25 smallest problems in Appendix A, and a 7-problem *training set* was selected at random among them, namely {d2,d4,d8,d10,d15,d18,d21}. F_{uq}, being the core for BIBD generation, was selected as the reference cost function, taking further advantage of the fact that it required no coefficient tuning. And DH (again, without parameters) was used as the basic reference strategy that would best represent the

inherent properties of the target space.

First, using DH over the training problem set, several cost function compositions were compared in an attempt to analyze the effects of each of the distribution measures. A strong interaction was observed between cost functions and problems (some functions performed best on particular problems, while other functions performed best on different problems, regardless of which distribution measures were involved). Yet, after trying about 60 different function compositions on each of the 7 problems, F_{uq} was found to perfom better overall: it was the one that yielded the smallest number of unsolved designs, and the largest number of significantly best scores (see [6] for details). Although not so clearly, second-best results were obtained with $F(\alpha_u, 1, 9, 3, 7, 0)$, termed hereafter F_{uthvq}, which was of particular interest for MBMUD generation [5].

Strategy parameters for SA and PMS were then tuned using F_{uq} over the training problem set (the choice of F_{uq} as the reference strategy had by then been reinforced by its good experimental results). For each strategy, experimental optimization was performed on each individual problem, leading to the *optimal* parameter setting for that problem, and the resulting values were then *generalized* (over problems and cost functions) to get a *standard* parameter setting for each strategy.

In the case of SA(T_c, ρ, N_t), a compromise temperature range was set to $\rho = 2$ (a wider range would mean a waste of relaxation effort, whereas a narrower range would make results too critical on the proper tuning of T_c) and optimization was perfomed over $(T_c \pm \Delta T_c, N_t \pm \Delta N_t)$, until the response at the central point was significantly best. After a case per case optimization, a good generalization criterion was found for T_c by normalizing temperatures with respect to optimal local increments, and the average optimal value for N_t was used. The *standard* parameter setting was thus selected to be

$$SA^{std} \equiv SA(0.110\Delta^0 F^*, 2, 100),$$

and it let to a small standardization loss.

The parameters for PMS(S, R_0, R_f, N_t) were tuned in a similar way. PMSfl, PMSfc and PMSs were treated independently, and results were best for PMSfl in a case per case basis. Therefore the other two variants were discarded. Being a new strategy, no process knowledge was available in the case of PMS, and optimization was more difficult. Four (instead of two) parameters were tuned, and results were often poor (including several occurrences of *nil* results), thus yielding larger deviations and making sometimes optimization ineffective. Optimal ranges were wider and imprecise and, for several problems, little sensitivity was found with respect to the radii R_0 and R_f. Since optimization was based on cost w, the cluster size S and the number of iterations N_t showed a marked tendency to optimize on small values. Problem d2 was also discarded

from the training set, since it optimized on $S = 1$ and $N_t = 0$ (i.e, a simple descent with DH). A good generalization of the optimal parameters could not be found, and the most-often best perfoming setting was chosen as a compromise for the standard PMS parameters:

$$PMS^{std} \equiv PMSfl(2, vb, 4, 15).$$

Actually, this was *not* the optimal setting for most of the training problems, and the standardization loss was large.

Finally, the performance of the selected cost functions and parameters was validated over different experimental settings. The comparison between F_{uq} and F_{uthvq} was extended to the two remaining strategies (using their standard parameters) and over the whole problem set. Although interaction was still present, F_{uq} consistenly performed best for *each* strategy, and therefore F_{uq} was selected for the comparison stage described in the next section. In the case of SA, the optimality of the standard parameters could also be verified over the whole test problem set, both for F_{uq} and F_{uthvq}. The standard parameters for PMS needed no furhter checking, since they were already known to be non-optimal.

4.3 Second Stage: Comparisons

During the comparison stage, the three standardized strategies DH, SA^{std} and PMS^{std} were compared over the *test* problem set using the best performing cost function F_{uq}. Results are shown in Table 2.

Except for problem d2, SA performed best in a case per case basis, and could solve the largest number of problems (all but d22). The comparison between DH and PMS is also particularly interesting, because it shows how the cooperative search effectively improved the results of independent Down-Hill search. In terms of the number of solved problems PMS, with only 3 cases unsolved (d19, d22 and d24), was clearly superior to DH, which failed on 12 occasions. But on the 13 cases solved by both, the score was 2 to 9 signifficant differences *against* PMS. Even tough PMS greatly improved efficacy, its efficiency was strongly penalized by its much larger descent cost. Yet, as shown in the previous section, PMS was actually handicapped by the standardization process, and might potentially yield much better performance if a better generalization could be found for its parameters. This assertion is supported by the results in Table 3, where both SA and PMS were run using their respective case-by-case optimal settings. Although SA is still best, PMS manages to get a star (on d18), and the score against DH turns 3 to 1 on its favor, over the 5 problems solved by both (notice that the difference on d10 is not significant).

Table 2
Response w for the three standard strategies with F_{uq} over the 7 problems in the training set (above) and the remaining 18 problems in the test set (below). Entries show means and deviations (within parenthesis) over 5 replications of the parallel experiment. For each problem, the entries signaled "$\star$ " show significantly best marks, and those signaled "$\circ$ " second best marks. When any of the replications is a *nil* result it is excluded from the mean, and the number in square brackets shows the actual number of replications averaged. Hyphens signal unsolved problems.

	DH		SA		PMS	
d2	$\star$ 7.3	(0.1)	$\circ$ 7.9	(0.2)	41.3	(0.5)
d4	926.6	(402.6)	$\star$ 80.0	(1.7)	$\circ$ 407.8	(41.7)
d8	-		$\star$ 61.9	(1.4)	$\circ$ 3641.9	(1374.4)
d10	$\circ$ 1095.5	(184.9)	$\star$ 572.4	(37.5)	3497.4	(472.7)
d15	164.8	(24.4)	$\star$ 41.1	(0.2)	193.0	(4.8)
d18	-		$\star$ 1906.8	(175.0)	$\circ$ 2988.4	(848.1)
d21	[1] 6443.5		$\star$ 421.9	(32.6)	$\circ$ [4] 43322.6	(21372.5)
d1	$\circ$ 13.2	(0.4)	$\star$ 7.3	(0.1)	40.5	(0.7)
d3	$\circ$ 16.3	(0.6)	$\star$ 13.0	(0.1)	59.8	(0.4)
d5	$\circ$ 149.4	(16.5)	$\star$ 112.3	(5.6)	482.2	(38.2)
d6	-		$\star$ 122.1	(1.6)	$\circ$ 737.2	(42.2)
d7	$\circ$ 215.5	(12.7)	$\star$ 48.2	(0.6)	502.5	(43.6)
d9	$\circ$ 765.1	(137.8)	$\star$ 233.9	(7.1)	2424.9	(403.5)
d11	$\circ$ 71.2	(5.6)	$\star$ 22.9	(0.4)	95.8	(2.6)
d12	$\circ$ [3] 9328.0	(93.8)	$\star$ 29441.2	(5128.6)	[1] 70806.0	
d13	-		$\star$ 453.7	(21.7)	$\circ$ 1630.3	(183.0)
d14	-		$\star$ 596.0	(13.5)	$\circ$ 9078.2	(5200.2)
d16	-		$\star$ 9381.1	(1507.9)	$\circ$ [1] 34308.0	
d17	-		$\star$ 3894.1	(512.9)	$\circ$ [1] 73872.0	
d19	-		$\star$ [1] 206995.0		-	
d20	-		$\star$ 840.3	(34.9)	$\circ$ 25615.7	(8779.6)
d22	-		-		-	
d23	-		$\star$ 3550.3	(372.2)	$\circ$ 42187.6	(24720.7)
d24	-		$\star$ 35200.7	(21245.0)	-	
d25	892.3	(418.6)	$\star$ 79.9	(2.1)	443.5	(32.8)

Table 3
Comparison between the three strategies using *optimal* parameters, using F_{uq} over the training problem set (15 replications). The optimal setting for d2 with PMS reduced to DH and it is thus excluded.

	DH		SA		PMS	
d2	⋆ 7.2	(0.1)	∘ 7.6	(0.2)		
d4	837.4	(254.9)	⋆ 79.0	(2.2)	∘ 163.8	(18.4)
d8	-		⋆ 59.4	(0.9)	∘ 1793.6	(266.0)
d10	1241.9	(318.4)	⋆ 644.6	(52.0)	1496.2	(268.3)
d15	162.4	(24.4)	⋆ 39.5	(1.0)	∘ 57.3	(4.0)
d18	-		∘ 1345.9	(63.0)	⋆ 646.6	(40.5)
d21	[7] 11094.8	(3155.9)	⋆ 345.7	(9.6)	∘ [10] 30373.6	(13519.1)

Results were then analyzed as a function of problem size, both in terms of efficacy $\mathbf{y}$ and efficiency $\mathbf{z}$. For the former, the estimated expected value $\hat{E}(\mathbf{y})$ varied considerably with problems and strategies, and the estimated deviations followed closely the expected values (as would be expected from a geometric distribution with a relatively low success probability). The precision of the estimates improved when the expected values were small. The evolution of the efficacy with problem size is shown in Figure 1 a), where the *inverse* estimator (the estimated frequency of success $\hat{p}$) is used for the sake of clarity. Comparing the three strategies, the qualitative shape of the curves is quite similar, showing that, although interaction is important, the difficulty of each individual problem has an intrinsic component. Although the evolution of the figure is erratic (the dependence on *vb* is not direct), the general tendency is quite clear: larger problems are more difficult. The superiority of SA is evident from the figure and, although the margin is not so wide, PMS outperforms DH in every case.

The cost z to the first solution is the main result of the analysis, and it measures the *efficiency* of the search. Deviations follow again expected values, and precision improves with good results. Since single descent cost deviations are small, the statistical behavior of $\mathbf{z}$ is dominated by the distribution of $\mathbf{y}$. The evolution of $\hat{E}(\mathbf{z})$ with problem size (Figure 1 b)) is again shown in terms of its inverse $10000/\mathbf{w}$, which can roughly be interpreted as the number of solutions that would be found within 10000 iterations. As before, the qualitative evolution of efficiency curves is similar for the three strategies but, now, the relative descent costs alter the relative scores among the strategies: DH shortens the distance to SA, and it outperforms PMS on all problems solved by both.

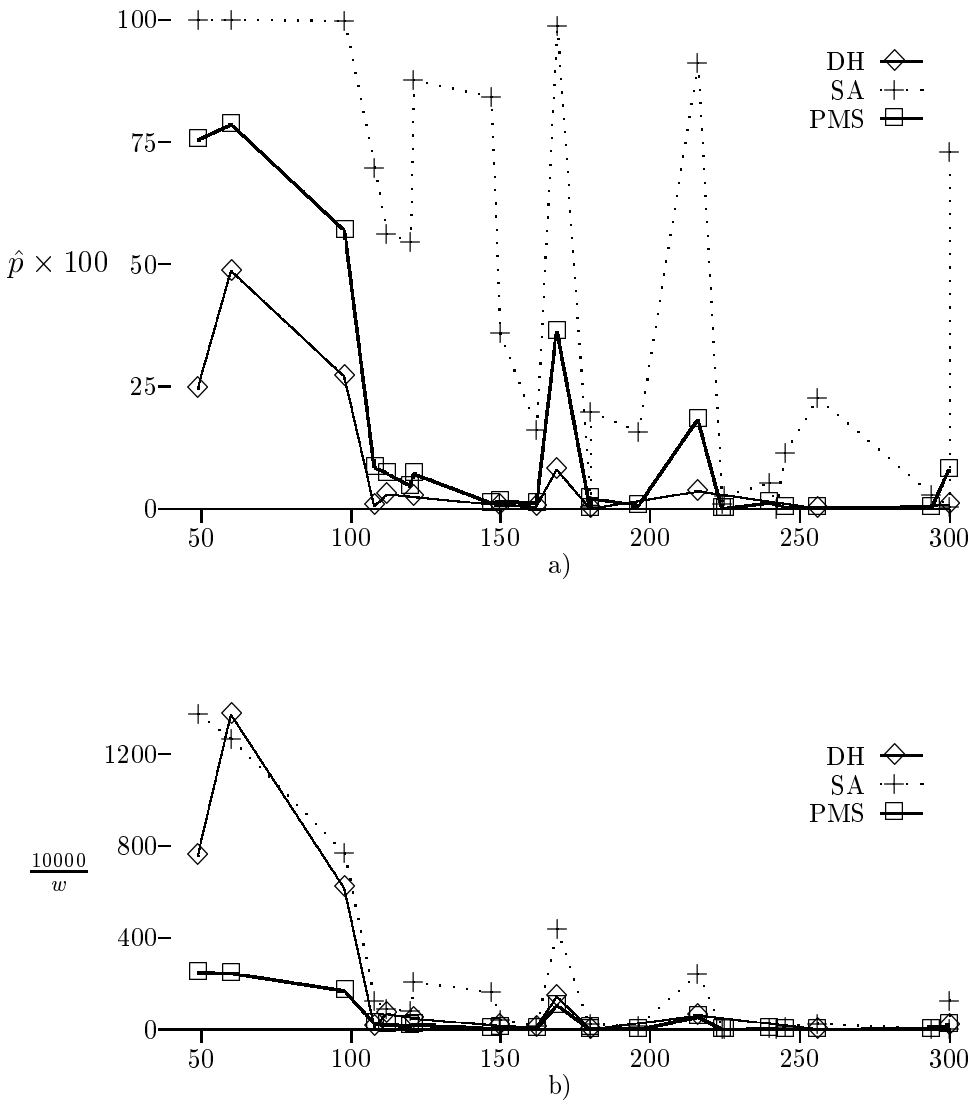

Fig. 1. For each of the standard strategies, with F_{uq}, a) probability of success $\hat{p} = 1/\hat{E}(\mathbf{y})$, and b) the inverse of the estimated cost to the first solution $10000/w$, as a function of problem size vb.

4.4 Third Stage: Problems of Growing Size

The last group of experiments were performed using the best algorithm of the previous stage (F_{uq} with SA^{std}). First, the search was applied to problems up to $vb \leq 500$ (d26 to d57 in Appendix A, excluding d27 and d55, which are known not to exist), and the estimated expected costs, after a single replication, are listed in Table 4. Out of 30 problems, 14 were unsolved, and the remaining 16 show very high cost values.

Finally, the remaining 70 problems with vb up to 1000 (d58 to d129 in Appendix A, excluding d71 and d107 as before), were attempted on a

Table 4
Estimated cost to the first solution for problems of size up to $vb = 500$, with the best algorithm of the comparison stage, F_{uq} with SA^{std} (1 replication).

	vb	$\hat{E}(\mathbf{z})$		vb	$\hat{E}(\mathbf{z})$
d26	300	8213.8	d42	392	25814.4
d28	320	7907.0	d43	396	206975.0
d29	324	451.4	d44	420	34409.3
d30	324	206807.0	d45	432	68942.3
d31	336	-	d46	441	85.5
d32	338	69080.7	d47	441	206799.0
d33	338	-	d48	448	68902.0
d34	338	1923.8	d49	448	-
d35	343	10291.8	d50	450	-
d36	360	-	d51	450	-
d37	360	17170.2	d52	480	-
d38	361	-	d53	480	206783.0
d39	363	-	d54	484	-
d40	364	-	d56	486	-
d41	384	-	d57	490	-

solved/unsolved basis only. As listed in Table 5, only 8 problems could be solved, the largest being d123, with $vb = 961$. A curious remark is that the two largest solved problems (d99 and d123) are the *only* ones in their group that are known to have exactly one non-isomorphic solution (see Appendix A).

From these results, although no regular behaviour can be found with respect to problem size, it is again made clear that larger problems are really much more difficult, as expected from an NP problem. Notice, as discussed in Section 1.1, that problems as large as these (like the unsettled d94 testifies) are already too large for exhaustive exploration.

5 Conclusions

In this work, the application of optimizing neural networks to the generation of block designs has been studied, leading to a theoretical characterization of the suitable cost functions, followed by an experimental comparison be-

Table 5
Solved/unsolved problems with *vb* up to 1000 entries.

	vb	Solved?		*vb*	Solved?		*vb*	Solved?
d58	507	-	d83	640	-	d108	845	-
d59	507	-	d84	640	-	d109	847	-
d60	512	-	d85	648	-	d110	864	-
d61	525	yes	d86	648	-	d111	882	-
d62	525	-	d87	675	-	d112	882	-
d63	528	yes	d88	676	yes	d113	882	-
d64	528	-	d89	676	-	d114	896	-
d65	529	-	d90	676	-	d115	900	-
d66	539	-	d91	720	-	d116	900	-
d67	540	-	d92	722	-	d117	900	-
d68	540	yes	d93	726	-	d118	918	-
d69	560	-	d94	726	-	d119	924	-
d70	578	-	d95	728	-	d120	945	-
d72	588	yes	d96	729	-	d121	960	-
d73	600	-	d97	735	-	d122	960	-
d74	600	-	d98	750	-	d123	961	yes
d75	605	yes	d99	750	yes	d124	961	-
d76	605	-	d100	756	-	d125	961	-
d77	605	-	d101	760	-	d126	968	-
d78	612	-	d102	768	-	d127	968	-
d79	625	-	d103	768	-	d128	972	-
d80	630	-	d104	792	-	d129	1000	-
d81	630	-	d105	792	-			
d82	637	-	d106	810	-			

tween networks implementing these functions together with different search strategies.

Using the cost to the first solution as the performance measure allowed for an objective comparison in terms of efficiency, and its estimation by means of the response of the parallel experiment (2048 simultaneous runs on a Connection Machine CM-200) was accurate enough, in spite of its statistical drawbacks

(occasional non-normality, unequal variances and *nil* results), that prevented the use of standard tests such as ANOVA. Problems, functions and strategies were the experimental factors that defined an experimental search space with infinitely many levels. Thus, a training stage for function selection and parameter tuning was required, on the basis of the basic strategy (Down-Hill) and the core cost function (F_{uq}), respectively. A high interaction between the experimental factors was observed and, although some choices were actually validated, a good parameter generalization could not always be found. In all, although results were consistent, experimental conclusions should not be extrapolated lightly.

Although individual problems were intrinsically hard or easy, the general trend was that larger problems were more difficult. Actually, most of the largest ones remained unsolved. Results could not be compared with other techniques in the literature, because most work in the combinatorics field is devoted to the unsettled cases, and no systematic result listings could be found in terms of generation cost.

The simplest cost function F_{uq} performed best for all search strategies, emphasizing the descriptive capabilities of quadruples, and the definition of BIBDs in terms of the measure Q. Yet, the generalized function definition proposed in this work, in terms of all the distribution measures, allows for different enhancements of the desired design properties. This could prove useful in cases where pseudo-optima with a particular structure might be of interest. The experimental analysis in terms of pseudo-optima (i.e, minimizing the average energy of local minima) is a line for further research.

The best quality of DH was its low descent cost, leading to a fairly good efficiency for the problems that it could solve. Yet, its efficacy was, as expected, the lowest (only 13 solved out of 25 problems in the test set). Nevertheless, DH was very useful as a reference, since the other two strategies were based on it.

Results with SA were the most efficient on a problem by problem basis, and only one problem (out of 25) remained unsolved. Process knowledge was useful in parameter optimization and standardization, and SA's experimental behavior was "friendly": good performance (leading to small deviations and good precision), few cases of *nil* results, and a very good generalization capacity, that could be validated against other cost functions and problems. Descent cost was effectively minimized, and the use of a smooth temperature schedule proved helpful. Using SA, most minima were already found by the end of the stochastic phase of the search.

Results with PMS were promising (22 out of 25 solved problems), showing the benefits of the parallel search, but its high descent cost penalized its efficiency.

Better results might be obtained, if only a way could be found of standardising its parameters, but the lack of process knowledge didn't help, and its experimental behaviour was less friendly: *nil* results were more frequent, deviations were larger (higher imprecision), and parameter sensitivity was lower. The variant PMSfl proved best. Since parameters were optimized on cost, the very smallest cluster size $S = 2$ yielded the best results. For PMS, minima didn't appear during the cooperative phase of the search, but formed on the (subsequent) independent down-hill relaxation. Thus, PMS can be interpreted as an effective way of improving the initial state for Down-Hill search. Notice finally that PMS and SA are not mutually exclusive. Since they are both based on the same principle yet they act on different mechanisms, *Parallel Mean Annealing*, the combination of the two, might be another interesting approach for further study.

The above comparison was done with general-purpose strategies. Building problem knowledge into the search might lead to better-performing, problem-specific algorithms. A possible approach along this line would be to do some sort of isomorphism reduction, in order to reduce search space. Analysis of the search space itself (ratio of local/optimal minima, attraction basins, structure of local minima, neighbourhood of optimal solutions, etc.) might provide useful hints for designing better strategies, and the characterization of problem hardness and optimal strategy parameters with respect to design parameters would also be interesting. A natural extension of Simulated Annealing is the deterministic Mean Field Annealing strategy. Even if no further reduction of the computational cost were obtained, the mean field model might allow for new and different ways to explore the extended continuous search space.

Summarizing, the comparison of algorithms has proven an intrinsically difficult task in itself, and BIBDs have proven a challenging problem for optimizing neural networks to solve. Yet, BIBDs constitute a rich and coherent collection of samples which permit making fine discriminations. Using them as a benchmark to test other optimizing techniques is one of our current research lines [21].

Appendix A Admissible parameter sets for BIBDs

Admissible parameter sets (v, b, u) with $vb \leq 1000$ (extracted from [19] and reordered by size). Problem number d_i, original numbering m_i, design parameters (v, b, u) and descriptors $[r, k, \lambda]$, problem size vb, and bounds on the number N_s of non-isomorphic solutions ($N_s = 0$, a design does not exist; $N_s =?$, unsettled case).

d_i	m_i	v	b	u	r	k	λ	vb	N_s
d1	m1	7	7	21	3	3	1	49	1
d2	m4	6	10	30	5	3	2	60	1
d3	m9	7	14	42	6	3	2	98	4
d4	m2	9	12	36	4	3	1	108	1
d5	m15	8	14	56	7	4	3	112	4
d6	m43	6	20	60	10	3	4	120	4
d7	m7	11	11	55	5	5	2	121	1
d8	m31	7	21	63	9	3	3	147	10
d9	m10	10	15	60	6	4	2	150	3
d10	m24	9	18	72	8	4	3	162	11
d11	m3	13	13	52	4	4	1	169	1
d12	m33	10	18	90	9	5	4	180	21
d13	m118	6	30	90	15	3	6	180	6
d14	m67	7	28	84	12	3	4	196	35
d15	m21	9	24	72	8	3	2	216	36
d16	m101	8	28	112	14	4	6	224	2224
d17	m20	15	15	105	7	7	3	225	5
d18	m236	6	40	120	20	3	8	240	13
d19	m47	11	22	110	10	5	4	242	4393
d20	m117	7	35	105	15	3	5	245	108
d21	m13	16	16	96	6	6	2	256	3

d_i	m_i	v	b	u	r	k	λ	vb	N_s
d22	m58	12	22	132	11	6	5	264	601
d23	m191	7	42	126	18	3	6	294	417
d24	m71	10	30	120	12	4	4	300	> 998
d25	m30	10	30	90	9	3	2	300	960
d26	m409	6	50	150	25	3	10	300	19
d27	m16	15	21	105	7	5	2	315	0
d28	m5	16	20	80	5	4	1	320	1
d29	m66	9	36	108	12	3	3	324	22521
d30	m150	9	36	144	16	4	6	324	≥ 12
d31	m278	8	42	168	21	4	9	336	≥ 943
d32	m23	13	26	104	8	4	2	338	2407
d33	m77	13	26	156	12	6	5	338	≥ 1017
d34	m8	13	26	78	6	3	1	338	2
d35	m276	7	49	147	21	3	7	343	≥ 9
d36	m195	10	36	180	18	5	8	360	≥ 1000
d37	m596	6	60	180	30	3	12	360	34
d38	m41	19	19	171	9	9	4	361	6
d39	m125	11	33	165	15	5	6	363	≥ 127
d40	m89	14	26	182	13	7	6	364	≥ 79
d41	m35	16	24	144	9	6	3	384	≥ 1512
d42	m357	7	56	168	24	3	8	392	≥ 35
d43	m56	12	33	132	11	4	3	396	$\geq 10^3$
d44	m816	6	70	210	35	3	14	420	48
d45	m145	9	48	144	16	3	4	432	≥ 330
d46	m6	21	21	105	5	5	1	441	1
d47	m477	7	63	189	27	3	9	441	≥ 10
d48	m275	8	56	168	21	3	6	448	≥ 101

d_i	m_i	v	b	u	r	k	λ	vb	N_s
d49	m524	8	56	224	28	4	12	448	≥ 2224
d50	m193	10	45	180	18	4	6	450	≥ 14819
d51	m109	15	30	210	14	7	6	450	≥ 11
d52	m130	16	30	240	15	8	7	480	$\geq 9 \times 10^7$
d53	m1078	6	80	240	40	3	16	480	76
d54	m247	11	44	220	20	5	8	484	≥ 4394
d55	m19	22	22	154	7	7	2	484	0
d56	m364	9	54	216	24	4	9	486	$\geq 10^6$
d57	m595	7	70	210	30	3	10	490	≥ 108
d58	m70	13	39	156	12	4	3	507	$\geq 10^3$
d59	m124	13	39	195	15	5	5	507	≥ 30
d60	m76	16	32	192	12	6	4	512	≥ 111
d61	m14	15	35	105	7	3	1	525	80
d62	m104	15	35	210	14	6	5	525	≥ 117
d63	m55	12	44	132	11	3	2	528	$\geq 10^6$
d64	m319	12	44	264	22	6	10	528	≥ 602
d65	m63	23	23	253	11	11	5	529	1103
d66	m735	7	77	231	33	3	11	539	≥ 107
d67	m480	10	54	270	27	5	12	540	$\geq 10^8$
d68	m235	9	60	180	20	3	5	540	≥ 330
d69	m819	8	70	280	35	4	15	560	≥ 2224
d70	m158	17	34	272	16	8	7	578	≥ 11
d71	m25	21	28	168	8	6	2	588	0
d72	m881	7	84	252	36	3	12	588	≥ 417
d73	m190	10	60	180	18	3	4	600	≥ 961
d74	m363	10	60	240	24	4	8	600	≥ 14819
d75	m116	11	55	165	15	3	3	605	≥ 436800

d_i	m_i	v	b	u	r	k	λ	vb	N_s
d76	m242	11	55	220	20	4	6	605	≥ 1
d77	m416	11	55	275	25	5	10	605	≥ 3337
d78	m179	18	34	306	17	9	8	612	$\geq 10^3$
d79	m40	25	25	225	9	9	3	625	78
d80	m102	15	42	210	14	5	4	630	≥ 103
d81	m49	21	30	210	10	7	3	630	≥ 414
d82	m1030	7	91	273	39	3	13	637	≥ 417
d83	m44	16	40	160	10	4	2	640	≥ 986
d84	m128	16	40	240	15	6	5	640	≥ 15
d85	m356	9	72	216	24	3	6	648	$\geq 10^7$
d86	m690	9	72	288	32	4	12	648	$\geq 10^6$
d87	m290	15	45	315	21	7	9	675	$\geq 10^8$
d88	m65	13	52	156	12	3	2	676	≥ 92714
d89	m149	13	52	208	16	4	4	676	≥ 2408
d90	m373	13	52	312	24	6	10	676	≥ 1018
d91	m891	10	72	360	36	5	16	720	$\geq 10^8$
d92	m208	19	38	342	18	9	8	722	≥ 7
d93	m608	11	66	330	30	5	12	726	$\geq 10^6$
d94	m78	22	33	264	12	8	4	726	?
d95	m451	14	52	364	26	7	12	728	≥ 80
d96	m98	27	27	351	13	13	6	729	≥ 7
d97	m131	21	35	315	15	9	6	735	$\geq 10^4$
d98	m599	10	75	300	30	4	10	750	≥ 29638
d99	m11	25	30	150	6	5	1	750	1
d100	m517	9	84	252	28	3	7	756	≥ 330
d101	m224	20	38	380	19	10	9	760	$\geq 10^{16}$
d102	m123	16	48	240	15	5	4	768	≥ 11

d_i	m_i	v	b	u	r	k	λ	vb	N_s
d103	m200	16	48	288	18	6	6	768	$\geq 10^8$
d104	m316	12	66	264	22	4	6	792	$\geq 10^3$
d105	m743	12	66	396	33	6	15	792	≥ 602
d106	m1088	9	90	360	40	4	15	810	$\geq 10^6$
d107	m28	29	29	232	8	8	2	841	0
d108	m241	13	65	260	20	4	5	845	$\geq 10^3$
d109	m826	11	77	385	35	5	14	847	$\geq 10^6$
d110	m683	9	96	288	32	3	8	864	$\geq 10^7$
d111	m46	21	42	210	10	5	2	882	≥ 10
d112	m75	21	42	252	12	6	3	882	≥ 1
d113	m259	21	42	420	20	10	9	882	≥ 4
d114	m284	16	56	336	21	6	7	896	≥ 1
d115	m476	10	90	270	27	3	6	900	$\geq 10^{12}$
d116	m888	10	90	360	36	4	12	900	$\geq 10^9$
d117	m537	15	60	420	28	7	12	900	$\geq 10^{18}$
d118	m176	18	51	306	17	6	5	918	≥ 3
d119	m293	22	42	462	21	11	10	924	≥ 2
d120	m280	15	63	315	21	5	6	945	≥ 2211
d121	m119	16	60	240	15	4	3	960	$\geq 6 \times 10^5$
d122	m618	16	60	480	30	8	14	960	$\geq 9 \times 10^7$
d123	m12	31	31	186	6	6	1	961	1
d124	m54	31	31	310	10	10	3	961	≥ 38
d125	m143	31	31	465	15	15	7	961	≥ 1266891
d126	m1095	11	88	440	40	5	16	968	$\geq 10^7$
d127	m108	22	44	308	14	7	4	968	≥ 1
d128	m880	9	108	324	36	3	9	972	≥ 330
d129	m160	25	40	400	16	10	6	1000	≥ 43

References

[1] Aarts E.H.L. and Korst J.H.M., "Boltzmann machines and their applications", *Proc. PARLE.* Springer-Verlag. Lecture Notes in Computer Science, Vol 258, p 34-50, 1987.

[2] Aarts E.H.L. and Korst J.H.M., *Simulated Annealing and Boltzmann Machines*, Wiley Interscience, 1988.

[3] Ackley D.H., Hinton G.E. and Sejnowski T.J., "A Learning Algorithm for Boltzmann Machines", *Cognitive Science*, Vol 9, 147, 1985.

[4] Bofill P., Fontdecaba E. and Torras C., "Optimization Networks for the Generation of Block Designs", *Journal of Artificial Neural Networks*, Vol 2(4), pp 302-312, 1995.

[5] Bofill P. and Torras C., "Neural Cost Functions for BIBDs leading to MBMUDs", submitted for publication.

[6] Bofill P. and Torras C., "Optimizing Neural Networks for the Generation of Block Designs", English version of the first author's PhD dissertation, *Technical Report* UPC-DAC-1997-76, November 1997.

[7] *CM-200 Technical Summary.* Thinking Machines Corporation. June 1991.

[8] Corneil D.G. & Mathon R.A., "Algorithmic Techniques for the Generation and Analysis of Strongly Regular Grafs and Other Combinatorial Configurations", *Ann. of Discrete Mathematics*, North Holland Publishing Company, Vol. 2, pp. 1-32, 1978.

[9] Gibbons P.B., "Computational Methods in Design Theory", *The CRC Handbook of Combinatorial Designs*, pp 730-740, 1996.

[10] Goles E. and Matamala M., "Dynamical and Complexity Results for High Order Neural Networks", *International Journal of Neural Systems*, Vol. 5(3), pp 241-252, 1994.

[11] Gutzmann K.M., "Combinatorial Optimization Using a Continuous State Boltzmann Machine" *Proceedings of IANN Conference*, San Diego, 1987.

[12] Hall M., *Combinatorial Theory*, Ed. John Wiley & Sons, Second Edition 1986.

[13] Hertz J., Krogh A., Palmer R.G., *Introduction to the Theory of Neural Computation*, Ed., Addison-Wesley, January 1993.

[14] Hopfield J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Nat. Academ. Sciences USA*, Vol 79, pp 2554-2558, 1982.

[15] Hopfield J.J. and Tank D.W., ""Neural" Computation of Decisions for Optimization Problems", *Byological Cybern.*, Vol. 52, pp. 141-152, 1985.

[16] John J.A and Mitchell T.J., "Optimal Incomplete Block Designs", *J. Roy. Statist. Soc. Ser. B*, Vol. 39, p 39-43, 1977.

[17] John van Rees G.H., "(r, λ)-designs", Colbourn C.H and Dinitz J.H. (Eds.) *The CRC Handbook of Combinatorial Designs*, CRC Press, p 434-436, 1996.

[18] Mathon R., Rosa A., "2-(v, k, λ) Designs of Small Order", *The CRC Handbook of Combinatorial Designs*, pp 3-41, 1996.

[19] Mathon R. and Rosa A., "Tables of parameters of BIBD with r$\leq$41 including existence, enumeration and resolvability results: an update", *Ars Combinatoria*, Vol 30, December, Winnipeg, Canada, 1990.

[20] McKay B.D. and Radziszowski S.P., "Towards Deciding the Existence of 2-(22,8,4) Designs", Journal of Combinatorial Mathematics and Combinatorial Computing, Vol. 22, pp. 211-22, 1996.

[21] Meseguer P. and Torras C., "Solving strategies for highly symmetric CSPs", Proc. Sixteenth Intl. Joint Conf. on Artificial Intelligence (IJCAI'99), Stockholm, Aug. 1999.

[22] Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., and Teller E., "Equation of State Calculations for Fast Computing Machines", *Journal of Chemical Physics*, Vol 21, pp 1087-1092. 1953.

[23] Montgomery D.C., *Design and Analysis of Experiments*, John Wiley & sons, third edition, 1991.

[24] Mullin C.R. and Gronau H.O.F., "PBDs and GDDs: The Basics", Colbourn C.H and Dinitz J.H. (Eds.) *The CRC Handbook of Combinatorial Designs*, CRC Press, p 185-193, 1996.

[25] Peterson C. and Anderson J.R., "A Mean Field Theory Learning Algorithm for Neural Networks", *Complex Systems*, Vol 1(5), 995-1019, 1987.

[26] Peterson C. & Södeberg B., "A New Method for Mapping Optimization Problems onto Neural Networks", *Int. Journ. Neural Sys.*, Vol. 1(1), pp. 3-22, 1989.

[27] Sejnowski T.J., "Higher-Order Boltzmann Machines", *Proc AIP*, Snowbird 1986.

[28] Street A. P. and Street D. J., *Combinatorics of Experimental Design*, Oxford Science Publications, Claredon, Oxford 1987.

[29] Street D.J. and Street A.P., "Partially Balanced Incomplete Block Designs", Colbourn C.H and Dinitz J.H. (Eds.) *The CRC Handbook of Combinatorial Designs*, CRC Press, p 419-423, 1996.

[30] Van den Berg J., "Neural Relaxation Dynamics, Mathematics and Physics of Recurrent Neural Networks with Applications in the Field of Combinatorial Optimization", *PhD thesis*, Erasmus University Rotterdam, 1996.