


## Article

# Evaluating Algorithm Efficiency for Optimizing Experimental Designs with Correlated Data

Lazarus K. Mramba <sup>1,2,\*</sup> and Salvador A. Gezan <sup>2</sup> <sup>1</sup> Department of Biostatistics, University of Kansas Medical Center, Kansas City, KS 66160, USA<sup>2</sup> School of Forest Resources and Conservation, University of Florida, Gainesville, FL 32611, USA; sgezan@ufl.edu

\* Correspondence: lmramba@kumc.edu

Received: 16 November 2018; Accepted: 12 December 2018; Published: 18 December 2018



**Abstract:** The search for efficient methods and procedures to optimize experimental designs is a vital process in field trials that is often challenged by computational bottlenecks. Most existing methods ignore the presence of some form of correlations in the data to simplify the optimization process at the design stage. This study explores several algorithms for improving field experimental designs using a linear mixed models statistical framework adjusting for both spatial and genetic correlations based on *A*- and *D*-optimality criteria. Relative design efficiencies are estimated for an array of algorithms including pairwise swap, genetic neighborhood, and simulated annealing and evaluated with varying levels of heritabilities, spatial and genetic correlations. Initial randomized complete block designs were generated using a stochastic procedure and can also be imported directly from other design software. Results showed that at a spatial correlation of 0.6 and a heritability of 0.3, under the *A*-optimality criterion, both simulated annealing and simple pairwise algorithms achieved the highest design efficiencies of 7.4% among genetically unrelated individuals, implying a reduction in average variance of the random treatment effects by 7.4% when the algorithm was iterated 5000 times. In contrast, results under *D*-optimality criterion indicated that simulated annealing had the lowest design efficiency. The simple pairwise algorithm consistently maintained highest design efficiencies in all evaluated conditions. Design efficiencies for experiments with full-sib families decreased with increasing heritability. The number of successful swaps appeared to decrease with increasing heritability and were highest for both simulated annealing and simple pairwise algorithms, and lowest for genetic neighborhood algorithm.

**Keywords:** genetic relationships; greedy algorithms; pairwise swap; simulated annealing; spatial correlations

## 1. Introduction

Generating field experimental designs often requires the experimental units to be replicated, randomized and apply some form of blocking to reduce heterogeneity. These properties ensure that results of an experiment are unbiased, optimal, and allow to perform appropriate inferences to a larger population [1]. For plant breeding, field trials are an important component that help to evaluate and select the genotypes (or treatments) with superior performance to be used as future parents or commercial varieties [2]. Breeding trials are often characterized by testing a large number of genetic entries with limited replication. The effect of these entries is often estimated by fitting a linear mixed model (LMM) that considers genotypes as a random effects, and that incorporates genetic relationships (or correlations) by a variance–covariance matrix obtained based on pedigree information or molecular markers.

Several proposed experimental designs exist for field trials, and these can be generated using widely available statistical software. However, the process of generating an optimal or near-optimal design, that maximizes the amount of information extracted with limited resources, is often ignored due to their intensive computational requirements, particularly for experimental designs with large number of treatments. Some authors have presented efficient procedures to construct experimental designs for breeding trials, including incomplete blocks, row-column and augmented designs (e.g., John and Williams [3] and Williams et al. [4]). However, these are mostly restricted to the assumption of fixed treatments effects, and therefore ignore the information provided by the genetic relationships. At the same time, it has been shown that modelling field spatial correlations (e.g., by incorporating an autoregressive residual structure) results in more efficient designs than assuming that residuals are independent and identically distributed [5,6]. Here, the framework of mixed models is advantageous over traditional linear models since they allow for specification of appropriate variance–covariance structures for both factors (e.g., genetic entries) and residuals (thorough spatial correlation), providing greater flexibility and more efficient downstream statistical analyses.

To generate experimental designs under the above framework an optimality criterion is used together with the implementation of an iterative search algorithm. *A*- and *D*-optimality information based criterion are the most widely used procedures in field experiments to generate optimal or near-optimal designs [7–9] and most recently were used by Butler et al. [6] and Mramba et al. [10] to design experiments with correlated observations. These procedures are very useful in the process of selecting an optimal design [11]. *A*-optimality criterion seeks to minimize the average variance of random treatment effects and can be expressed as:  $A_{\text{optim}} = \operatorname{argmin}\{\operatorname{trace}[\mathbf{M}(\Omega)]\}$ , where  $\mathbf{M}(\Omega)$  is the inverse of an information matrix of the treatment (or genetic) effects from a given design layout  $\Omega$ . *D*-optimality was introduced by Wald [12] and minimizes the determinant of  $\mathbf{M}(\Omega)$  which can be interpreted as minimizing the generalized variance of the treatment effects [11] by choosing designs which minimize the volume of the joint confidence ellipsoid [13] and is given by  $D_{\text{optim}} = \operatorname{argmin}\{|\mathbf{M}(\Omega)|\}$  for  $|\mathbf{M}(\Omega)| \neq 0$ .

Often, search algorithms involve interchanging the assigned treatments for a pair of experimental units and re-evaluating the efficiency of the new design to be compared against the previous one. Some of the computer search algorithms available include pairwise swap procedure, and its variants where a single or multiple pairs of treatments are swapped at a time [3], and simulated annealing where a cooling strategy is employed [14]. Most of the applications of these algorithms focus on the analysis of data and little has been done on their applications to improve the designs of genetic experiments, yet, estimated parameters from improved designs can be obtained with increased precision if variability of treatment effects is minimized [10].

Although there are statistical software such as CycDesign [15], GenStat [16], SAS [2] and DiGger [17]; these programs are not freely available and do not account for both spatial and genetic relatedness of experiments at the design stage. The focus of the present study is to evaluate performance of different algorithms based on a linear mixed model framework which optimally accounts for both sources of correlation in an experiment at the design stage. Hence, the main objective of this study is to evaluate the efficiency of diverse search algorithms to generate improved randomized complete block (RCB) designs applying *A*- or *D*-optimality criteria, while accounting for both spatial and genetic correlations using linear mixed models with applications in plant breeding trials. This will be done by initially generating experimental layouts through a random process and later applying an array of proposed search algorithms to improve the initial experimental layouts. The procedure also allows optimizing designs initially generated from other software. Several varying field conditions that include a range of heritabilities, genetic relatedness structures and spatial correlations were evaluated in order to thoroughly assess the practicality of the algorithms presented. An illustration, given in Section 2.3, describes a practical example where the inclusion of a microsite random error, also known as the nugget effect or unstructured residual error is provided. The importance of including a nugget effect has been previously noted on other studies such as Cressie [18] and Gezan et al. [5] where the

latter study showed that in modeling spatial data, omission of the nugget error could lead to a bias in the correlation parameters of the error structure. Hence, the nugget error component could be used successfully to model potential microsite variability between observations that are closely spaced.

## 2. Materials and Methods

### 2.1. Statistical Model for Randomized Complete Block Designs

The linear mixed model (LMM) framework for RCB designs can be expressed as  $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{g} + \mathbf{e}$ , where  $\mathbf{y}$  is a vector of observed phenotypes (responses);  $\mathbf{X}$  is an incidence matrix of fixed block effects;  $\boldsymbol{\beta}$  is a vector of fixed block effects;  $\mathbf{Z}$  is an incidence matrix of random treatment effects;  $\mathbf{g}$  is a vector of random treatment effects, with  $\mathbf{g} \sim MVN(\mathbf{0}, \mathbf{G})$ , where  $\mathbf{G} = \sigma_g^2 \mathbf{A}$  for genetically correlated observations, with  $\mathbf{A}$  being the numerator relationship matrix calculated from pedigree information or molecular markers to account for additive genetic relatedness between individuals and  $\mathbf{G}$  is a variance–covariance matrix for genetic relationships. For instance,  $\mathbf{G} = \sigma_g^2 \mathbf{I}$  for genetically unrelated individuals. The vector  $\mathbf{e}$  represents residual errors, with  $\mathbf{e} \sim MVN(\mathbf{0}, \mathbf{R})$ , where  $\mathbf{R}$  is a variance–covariance matrix for modelling correlated errors. Most often,  $\mathbf{R}$  is modelled with an autoregressive error structure of order 1 [19]. To obtain the variance–covariance matrix of random treatment effects, linear mixed model normal equations are solved as described by Henderson [20] and Hooks et al. [21] to give

$$\mathbf{M}(\Omega) = Var(\hat{\mathbf{g}} - \mathbf{g}) = (\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} - \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X}(\mathbf{X}'\mathbf{R}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{R}^{-1}\mathbf{Z})^{-1} \quad (1)$$

from which the trace and determinant of the matrix  $\mathbf{M}(\Omega)$  are calculated based on *A*- and *D*-optimality criteria, respectively (For further details see [10]).

### 2.2. Algorithms

The algorithms implemented to generate improved designs are:

1. Simple Pairwise (SP), that swaps a single pair of treatments at a time,
2. Greedy Pairwise (GP), that swaps more than a single pair of treatments at a time,
3. Genetic Neighbourhood (GN), that takes into consideration the genetic relatedness of the direct neighbouring of a experimental units to perform swaps, and
4. Simulated Annealing (SA), that swaps a pair of treatments at a time, but accepts poor designs at random with a given probability which diminishes with time.

The procedure for these algorithms involves randomly generating  $m$  initial experimental layouts, denoted as  $\Omega_i$ . For each layout, the variance–covariance matrix of the treatment effects  $\mathbf{M}(\Omega)$  is obtained, and its criterion value is calculated (as trace or determinant (or log(determinant)) with *A*- and *D*-optimality, respectively). Next, from the  $m$  designs, the “best” experimental layout is selected, where “best” refers to the design with the smallest trace under *A*-optimality and design with the largest determinant under the *D*-optimality. After this, an optimization algorithm is applied for  $p$  iterations. For all implemented algorithms, the output is a list of objects including the improved experimental layout, a vector with criterion values and iterations of the sequentially accepted (successful) designs, and a vector of all criterion values from all iterations, whether the swap was successful or not. Following is a description of the implemented algorithms.

For the SP algorithm, the following steps are undertaken after selecting the best initial  $\Omega_i$  with criteria value  $\tau_i$ : (1) randomly interchange a single pair of treatments within a randomly selected block to produce a new layout,  $\tau_j$ ; (2) re-calculate a new criterion value  $\tau_j$ ; (3) if  $\tau_i > \tau_j$ , accept  $\Omega_j$  as the new layout; and (4) repeat steps 1 to 3 for a total of  $p$  iterations and produce the output.

GP algorithms are a more aggressive variant of the simple pairwise algorithm (SP) that allow multiple treatments to be randomly interchanged within a block. In order to evaluate a spectrum of

alternative implementations, this algorithm was implemented by varying the number of treatments to be swapped simultaneously, denoted as  $G\alpha$ , where  $\alpha$  refers to the number of treatments swapped. The implemented algorithm allows specification of any even number of treatments to be swapped at a time. Tested procedures were denoted as GP4, GP14 and GP98 for randomly swapping 4, 14 and 98 treatments simultaneously on each iteration within a randomly selected block, respectively. Numbers 14 and 98 were chosen as a percentage ( $\approx 50\%$ ) of the treatments to be swapped at a time, in an experiment with 30 and 196 treatments, respectively, whereas 4 was chosen as a close value to 2 to detect any small changes in improvement of the design when a single pair or double pairs of treatments are swapped in each iteration. Steps 1 to 4 apply as described under the SP procedure.

The GN algorithm is defined as a method that makes use of genetic relatedness of the eight neighbouring experimental units found in a  $3 \times 3$  matrix using information provided by the numerator relationship matrix ( $\mathbf{A}$ ) of the corresponding genotypes. Steps for this algorithm are: (1) randomly generate  $m$  initial designs and select the best ( $\Omega_i$ ) with the smallest trace,  $\tau_i$ ; (2) randomly select a treatment  $t_l$  from  $\Omega_i$ ; (3) identify the genetic correlation coefficients from the numerator relationship matrix for all experimental units within the nearest neighborhood of  $t_l$ ; (4) if there exists a pairwise genetic relationship of 0.25 or higher between  $t_l$  and any other treatment  $t_k$  for  $l \neq k$  within the neighbouring matrix, then replace either one of the treatments with a another treatment that is at a distance of more than a unit (row or column) away; (5) if there are no treatments further than a unit away even though these neighbours are genetically correlated, randomly interchange  $t_l$  with  $t_k$ ; (6) calculate the new criterion value,  $\tau_j$ , based on the new design layout  $\Omega_j$ ; (7) if  $\tau_i > \tau_j$ , accept  $\Omega_j$ , otherwise reject  $\Omega_j$ ; and (8) repeat steps 2 to 7 for a total of  $p$  iterations. Note that if all the experimental units from a neighbourhood are genetically unrelated, then the SP is applied.

SA is a probabilistic meta-heuristic and stochastic optimization procedure that prevents the search from getting trapped in a local optima by accepting some solutions with a set probability and lowering the temperature with time to make sure that poorer solutions are accepted with lower probabilities [14]. The SA algorithm implemented in this study is described as follows: (1) randomly interchange a pair of treatments within a randomly selected block to produce a new layout,  $\Omega_j$  and re-calculate a criterion value,  $\tau_j$ ; (2) if  $\tau_i > \tau_j$ , accept  $\Omega_j$  as the new layout with probability 1.0; else do the following step, (3) calculate  $\Delta = \tau_j - \tau_i$  and set a cooling temperature  $T_c[i] = 1/i$ , for the  $i$ -th iteration, and calculate  $v = \exp(-\Delta / T_c[i])$ ; (4) draw a random value  $u$  from a uniform distribution, and if  $u < v$  accept  $\Omega_j$ ; and (5) repeat steps 1 to 4 for a total of  $p$  iterations.

The SA method has two parameters that have to be tuned, i.e., initial temperature and cooling rate. In this study, the initial temperatures are the initial criterion values (that is, traces or determinants) of the initial designs before optimization process. The starting initial temperature was chosen to be the best criterion value among the initial designs. For instance, under the  $A$ -optimality criterion, choose a design that has the smallest trace (equivalent to starting temperature) as the main initial design to be optimized further. The cooling rate was viewed as part of step 3 above, and also, it can be viewed as a stopping rule. For example, the stopping rule can be the difference between the current criterion value and the previously calculated value (say, a difference of 0.05) observed for a consecutive number of iterations. The stopping rule for the motivating example was set to be the number of iterations  $p = 20,000$  and for all other illustrations, it was set to  $p = 5000$  iterations.

### 2.3. Evaluation of Algorithms

The above four algorithms were evaluated under varying experimental conditions to assess their effectiveness to improve field designs. Conditions considered include narrow-sense heritabilities,  $m = 1$  initial designs,  $h^2$ , of 0.1, 0.3, and 0.6, where  $h^2 = \sigma_g^2 / (\sigma_g^2 + \sigma_e^2)$ ; unrelated individuals (independent), half-sib and full-sib families; and a spatial correlation of  $\rho = 0.6$ . Every combination of conditions was repeated  $\lambda = 10$  times for  $p = 5000$  iterations. All implementation and evaluation of algorithms was done using the statistical package R [22].

The following scenarios were considered:  $\Omega_A^{(30)}$ ,  $\Omega_D^{(30)}$  and  $\Omega_A^{(196)}$ . The first two represent RCB designs with 30 genotypes generated using  $A$ - and  $D$ -optimality criteria, respectively. In these layouts, the designs had six blocks each of dimensions five rows by six columns. Here, pedigree from half-sib families consisted of five male parents each with six individuals, and full-sib families consisted on a half-diallel with five parents for a total of 10 families each with three individuals. Scenario  $\Omega_A^{(196)}$  represents an RCB design with 196 genotypes generated using  $A$ -optimality criterion with four blocks of dimensions 14 rows by 14 columns per block. Pedigree files for half-sib families had 32 known parents each with six offspring, whereas full-sib families had 30 parents with several half-diallels for a total of 68 families each with approximately three offspring. Note that, GP98 was implemented only for  $\Omega_A^{(196)}$  scenario to swap 50% of the total genotypes at every single iteration, and GP14 represents swapping about 50% of the genotypes for  $\Omega_A^{(30)}$  and  $\Omega_D^{(30)}$  scenarios.

A detailed practical example was implemented with all algorithms in order to investigate the level of design efficiencies and rates of convergence that can be obtained for a specified condition with all algorithms having to improve the same initial RCB experimental design. This was done using  $A$ -optimality criterion for an experiment with 30 genotypes, 6 blocks of sizes 5 rows by 6 columns, and comprised of half-sib families with five male parents each with six individuals, for  $h^2 = 0.1$ ,  $\rho = 0.6$ , and an arbitrary nugget effect of 0.1. Initially,  $m = 1000$  designs were randomly generated and the best one selected for optimization. All the proposed algorithms were made to improve this initial design by going through  $p = 20,000$  iterations. Traces from both successful and unsuccessful swaps were observed together with the time taken for each algorithm. The practical example was run from a 64-bit windows operating system Intel(R) Core(TM) i7-4720HQ CPU@2.60GHz, RAM 8.0 GB.

To evaluate the improvement of a design, relative overall design efficiency (ODE), that quantifies how efficient the improved design is relative to an initially non-improved design for  $A$ - and  $D$ -optimality was calculated as a proportion or percentage difference between the initial best criterion and the final optimal-value:

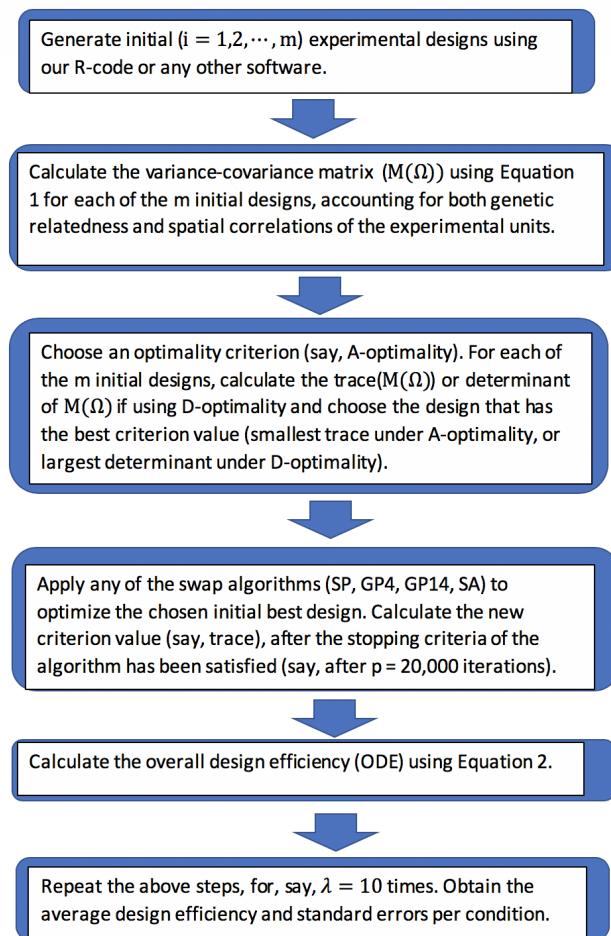
$$\gamma_{ij}^A = \frac{\bar{A}_{ij} - A_{(opt)ij}}{\bar{A}_{ij}}; \quad \gamma_{ij}^D = \frac{\bar{D}_{ij} - D_{(opt)ij}}{\bar{D}_{ij}}; \quad (2)$$

for  $i = 1, 2, \dots, \xi$  conditions;  $j = 1, 2, \dots, \lambda$  replicates

where  $\bar{A}_{ij}$  and  $\bar{D}_{ij}$  are averages of  $m$  initial traces and log-determinants, respectively, for  $i$ -th condition and  $j$ -th replicate,  $A_{(opt)ij}$  and  $D_{(opt)ij}$  are the smallest trace and log-determinant, respectively, obtained from an improved design. Finally, ODE calculations over the  $\lambda = 10$  replicates per condition were summarized. A schematic diagram that represents a summary of the procedure to improve a given randomized complete block design is displayed in Figure 1 and Table 1 shows the simulation conditions implemented for the motivating example.

The R-code that was used for the algorithms described in this paper have been provided in Appendix A. The R-code to generate the initial RCBD before optimization is shown in Appendix B while Appendices C and D provide the R-code for generating the numerator relationship matrix and the variance-covariance matrix, respectively. Supplementary materials that include additional R-code, a worked-out example using RMarkdown and the pedigree information are also available for illustration purposes.





**Figure 1.** Schematic diagram to summarize the procedure for improving a randomized complete block (RCB) design.

**Table 1.** Simulation conditions for the motivating example assuming a spatial correlation  $\rho = 0.6$  and a nugget error of 0.1 for the three designs:  $\Omega_A^{(30)}$ ,  $\Omega_A^{(196)}$  and  $\Omega_D^{(30)}$ , where  $\Omega_A^{(30)}$  represents an RCB design with 30 treatments (genotypes) arranged in 6 blocks of sizes 5 rows by 6 columns and optimized using an A-optimality criterion. Initial  $m = 1000$  designs were generated and the overall best design (design with smallest trace under A-optimality or largest determinant under D-optimality) selected to be optimized. The algorithm was stopped after  $p = 20,000$  iterations. Final designs for each condition represented the improved design and the ODE % are calculated using Equations (1) and (2). Each of the nine conditions was repeated  $\lambda = 10$  times for all four algorithms: SP, GP4, GP14 and SA.

Condition	$h^2$	Pedigree
1	0.1	Indep
2	0.3	
3	0.6	
4	0.1	Half-sib
5	0.3	
6	0.6	
7	0.1	Full-sib
8	0.3	
9	0.6	

### 3. Results

Averages and standard errors (S.E.) of overall design efficiency (ODE %) for the three scenarios, that is,  $\Omega_A^{(30)}$ ,  $\Omega_A^{(196)}$  and  $\Omega_D^{(30)}$  for all algorithms are presented in Tables 2–4, respectively. Figure 2 displays visible trends of ODEs by genetic relatedness and heritability levels whereas Figure 3 shows the average number of successful swaps out of 5000 (that is, swaps that were accepted due to the resulting design having a smaller criterion value than the previous layout) for each algorithm. These results indicate that, for all experiments conducted based on  $\Omega_A^{(30)}$  and  $\Omega_A^{(196)}$  scenarios, simulated annealing (SA) and simple pairwise (SP) algorithms achieved the highest ODE averages in all evaluated conditions followed by GP4 (for  $\Omega_A^{(30)}$ ) or GP98 (for  $\Omega_A^{(196)}$ ) and lowest for genetic neighbourhood (GN). Also, the overall highest ODEs were achieved when  $h^2 = 0.3$  among genetically unrelated individuals for all algorithms. Among full-sib families, highest ODEs were achieved when  $h^2 = 0.1$  and decreased with increasing heritability for all algorithms evaluated under  $\Omega_A^{(30)}$  and  $\Omega_A^{(196)}$  scenarios. SA recorded the highest average ODE of 7.403% (S.E. = 0.063) followed by SP with average ODE of 7.398% (S.E. = 0.066) all obtained when  $h^2 = 0.3$  among genetically unrelated individuals. Algorithms SA, SP, GP4 and GP14 evaluated with half-sib families under  $\Omega_A^{(30)}$  had highest ODEs obtained for treatments with the lowest heritability of 0.1, whereas GN achieved its highest ODE when  $h^2 = 0.3$  for the same genetic structure.

Based on a  $\Omega_D^{(30)}$  scenario, the best performing algorithm with highest average ODE among all conditions was SP, closely followed by GP4, GP14, GN and SA which recorded the lowest average ODE. Under this scenario, the overall highest ODEs were observed among genetically unrelated individuals for SP, GP4, and GP14 when  $h^2 = 0.3$ . Among half-sib families, highest ODEs occurred when  $h^2 = 0.3$  but no clear trends among full-sib families were observed.

Both  $\Omega_A^{(30)}$  and  $\Omega_D^{(30)}$  took, on average, about 2 min to improve a given initial experimental design for  $p = 5000$  iterations, whereas  $\Omega_A^{(196)}$  required about 25 min for the same number of iterations. Figure 3 shows that the number of successful swaps decrease with increasing heritability especially for  $\Omega_A^{(30)}$  and  $\Omega_A^{(196)}$  scenarios with small difference in numbers between SA and SP algorithms but larger differences are noted under  $\Omega_D^{(30)}$  scenario. The number of successful swaps out of 5000 appeared to be highest for SA and SP under A-optimality criterion. From  $\Omega_D^{(30)}$  scenario, the number of successful swaps were highest for SA which recorded above 2500 out of the 5000 swaps but this was not reflected in terms of improving the overall design efficiency under this criterion in contrast with other algorithms.

**Table 2.** Average ODEs from 10 replicates per condition are reported together with standard errors (S.E.) for simple pairwise (SP), greedy pairwise (GP4 and GP14), simulated annealing (SA) and genetic neighborhood (GN) procedures for  $\Omega_A^{(30)}$  RCB designs at a spatial correlation of 0.6.

Condition		SP		GP4		GP14		SA		GN	
Pedigree	$h^2$	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.
Indep	0.1	6.347	0.060	5.501	0.060	3.747	0.093	6.385	0.072	-	-
	0.3	7.398	0.066	6.194	0.080	4.371	0.053	7.403	0.063	-	-
	0.6	5.109	0.044	4.414	0.057	3.110	0.054	5.222	0.064	-	-
Half-sib	0.1	5.826	0.026	5.082	0.055	3.610	0.065	5.781	0.045	1.853	0.042
	0.3	5.375	0.056	4.640	0.082	3.192	0.052	5.428	0.047	1.940	0.088
	0.6	3.066	0.028	2.663	0.023	1.858	0.033	3.131	0.028	1.064	0.033
Full-sib	0.1	4.109	0.030	3.611	0.026	2.543	0.038	4.045	0.027	1.343	0.034
	0.3	2.656	0.029	2.265	0.021	1.601	0.034	2.667	0.032	0.920	0.027
	0.6	1.247	0.006	1.065	0.009	0.755	0.012	1.247	0.013	0.460	0.011

**Table 3.** Average ODEs reported together with standard errors (S.E.) for simple pairwise (SP), greedy pairwise (GP4 and GP98), simulated annealing (SA) and genetic neighborhood (GN) procedures for  $\Omega_A^{(196)}$  RCB designs at a spatial correlation of 0.6.

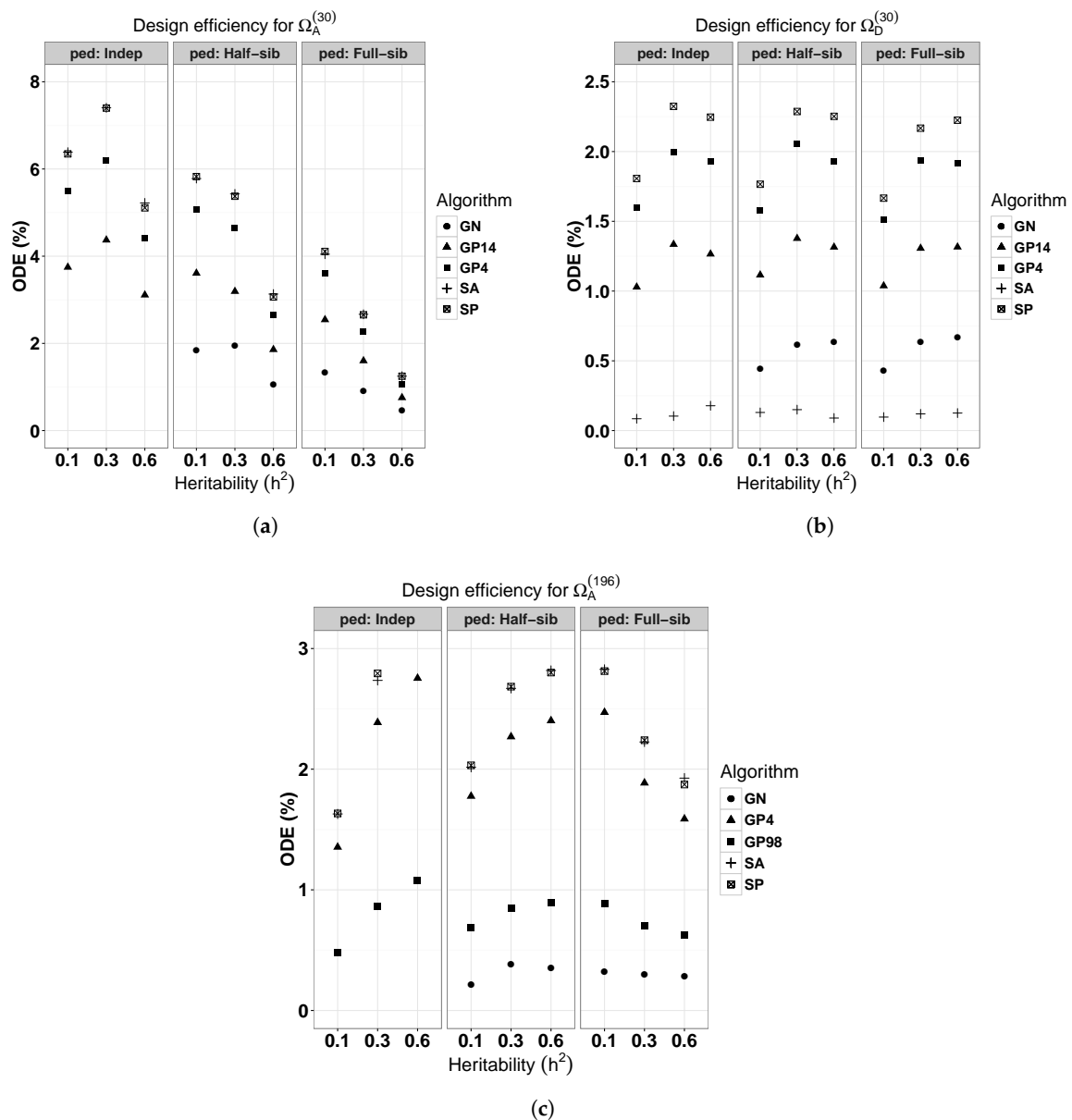
Condition		SP		GP4		GP98		SA		GN	
Pedigree	$h^2$	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.
Indep	0.1	1.633	0.013	1.354	0.018	0.481	0.008	1.629	0.015	-	-
	0.3	2.794	0.020	2.387	0.017	0.864	0.024	2.736	0.034	-	-
	0.6	3.232	0.024	2.754	0.039	1.080	0.028	3.270	0.027	-	-
Half-sib	0.1	2.032	0.023	1.776	0.019	0.690	0.018	2.016	0.019	0.216	0.014
	0.3	2.684	0.018	2.269	0.009	0.851	0.024	2.670	0.019	0.381	0.013
	0.6	2.801	0.027	2.402	0.029	0.890	0.025	2.818	0.009	0.351	0.020
Full-sib	0.1	2.813	0.018	2.471	0.022	0.888	0.025	2.827	0.014	0.324	0.015
	0.3	2.240	0.016	1.886	0.023	0.702	0.020	2.226	0.021	0.297	0.011
	0.6	1.873	0.011	1.588	0.013	0.623	0.016	1.926	0.013	0.280	0.013

**Table 4.** Average ODEs reported together with standard errors (S.E.) for simple pairwise (SP), greedy pairwise (GP4 and GP14), simulated annealing (SA) and genetic neighborhood (GN) procedures for  $\Omega_D^{(30)}$  RCB designs at a spatial correlation of 0.6.

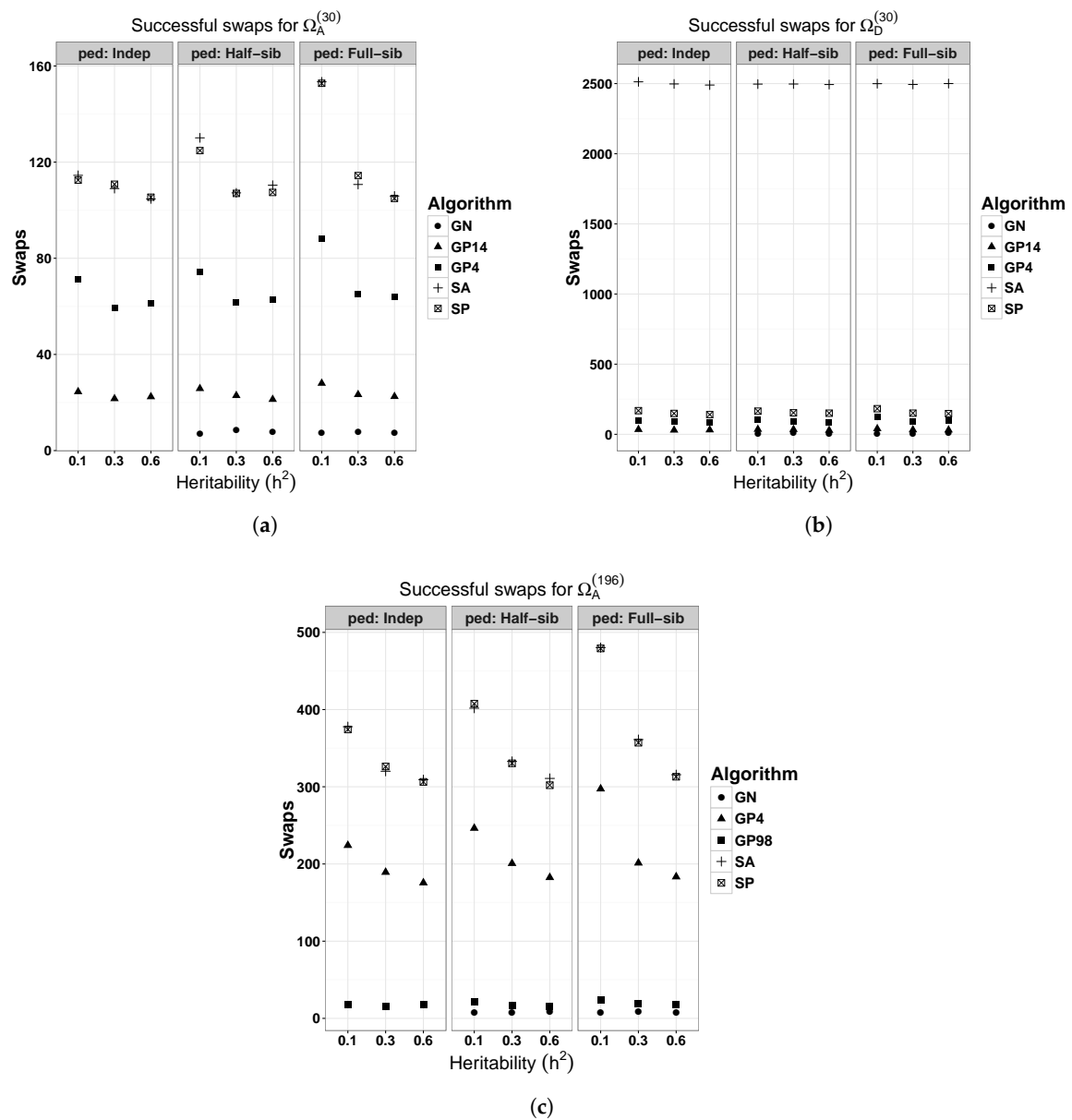
Condition		SP		GP4		GP14		SA		GN	
Pedigree	$h^2$	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.	ODE %	S.E.
Indep	0.1	1.807	0.014	1.600	0.014	1.029	0.014	0.085	0.019	-	-
	0.3	2.324	0.017	1.993	0.013	1.335	0.030	0.104	0.025	-	-
	0.6	2.247	0.021	1.930	0.021	1.265	0.032	0.178	0.027	-	-
Half-sib	0.1	1.766	0.012	1.576	0.015	1.115	0.015	0.130	0.034	0.446	0.015
	0.3	2.287	0.023	2.054	0.024	1.377	0.022	0.150	0.041	0.614	0.013
	0.6	2.253	0.024	1.933	0.020	1.315	0.019	0.090	0.023	0.637	0.023
Full-sib	0.1	1.666	0.011	1.514	0.013	1.037	0.009	0.097	0.025	0.431	0.010
	0.3	2.168	0.013	1.935	0.025	1.307	0.026	0.119	0.025	0.634	0.017
	0.6	2.225	0.027	1.913	0.023	1.316	0.025	0.125	0.019	0.669	0.022

Results from the practical example that was conducted for an RCB design with  $h^2 = 0.1$  and  $\rho = 0.6$  based on  $\Omega_A^{(30)}$  are displayed in Figure 4 which plots traces obtained from successful swaps and their overall design efficiencies. Also, Figure 5 shows the rate of convergence by plotting all the 20,000 traces obtained for each algorithm. From this illustration, the results indicate that the SP algorithm had the highest design efficiency of 6.713% with the highest number of successful swaps (192) and took about 5.8 min for the 20,000 iterations. This was closely followed by the SA algorithm that had an ODE of 6.258% with 139 successful swaps and took about 5.8 min. GP4 algorithm had an ODE of 5.552% with 104 successful swaps and also took about 5.8 min and finally, the GN algorithm recorded the lowest ODE of 2.053% with 12 successful swaps and took about 6.1 min.

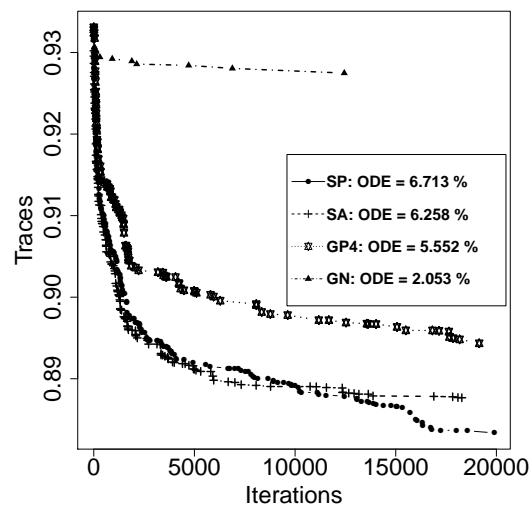




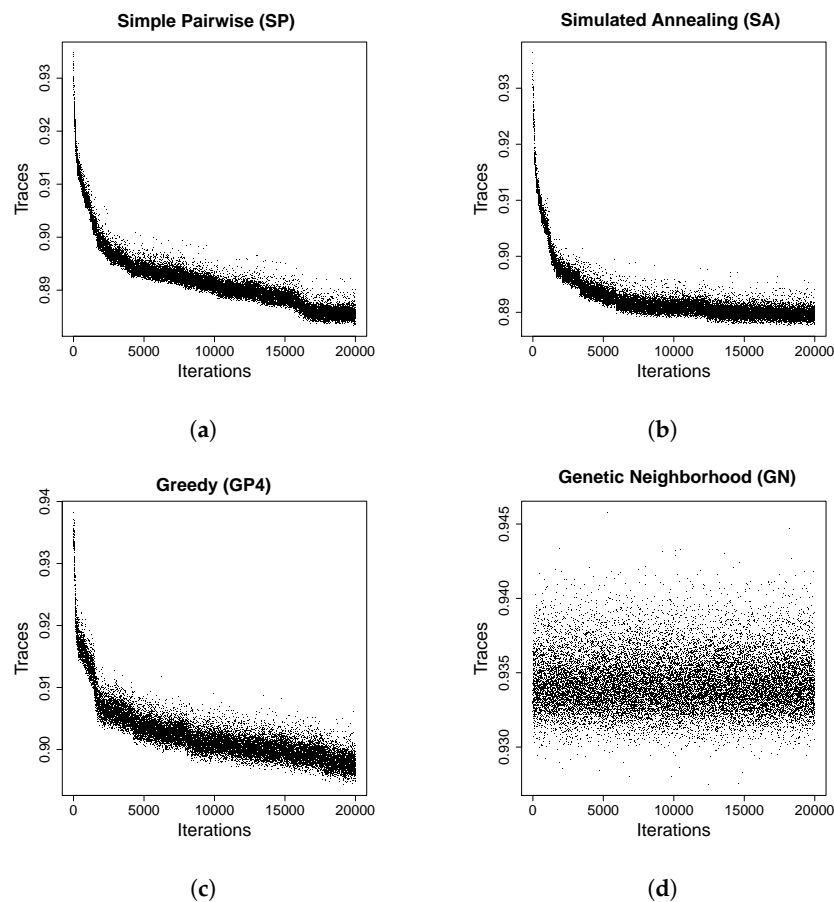
**Figure 2.** Overall design efficiency (ODE %) for (a)  $\Omega_A^{(30)}$ , (b)  $\Omega_D^{(30)}$ , and (c)  $\Omega_A^{(196)}$  scenarios evaluated for simple pairwise (SP), greedy pairwise (GP4, GP14, GP98), simulated annealing (SA) and genetic neighborhood (GN) algorithms iterated  $p = 5000$  times, with each condition replicated  $\lambda = 10$  times, with  $m = 100$  initially unimproved designs.



**Figure 3.** Average number of swaps for (a)  $\Omega_A^{(30)}$ , (b)  $\Omega_D^{(30)}$ , and (c)  $\Omega_A^{(196)}$  scenarios evaluated for simple pairwise (SP), greedy pairwise: GP4, GP14, GP98, simulated annealing (SA) and genetic neighborhood (GN) algorithms iterated  $p = 5000$  times, with each condition replicated  $\lambda = 10$  times, with  $m = 100$  initially unimproved designs.



**Figure 4.** Traces from successful swaps to convey the rate of convergence for simple pairwise (SP), simulated annealing (SA), greedy pairwise (GP4), and genetic neighborhood (GN) algorithms with their overall design efficiencies (ODE) evaluated for half-sib families with  $h^2 = 0.1$ ,  $\rho = 0.6$  and a nugget error of 0.1 iterated for 20,000 for a practical  $\Omega_A^{(30)}$  example.



**Figure 5.** Rates of convergence for (a) simple pairwise (SP), (b) simulated annealing (SA), (c) greedy pairwise (GP4), and (d) genetic neighborhood (GN) showing all traces obtained from these algorithms evaluated for half-sib families with  $h^2 = 0.1$ ,  $\rho = 0.6$  and a nugget error of 0.1 iterated for 20,000 for a practical  $\Omega_A^{(30)}$  example.

#### 4. Discussion

Optimization algorithms are commonly used in agriculture and forestry research for several planning and management problems [23,24]. In the current study, an array of these algorithms was implemented and evaluated to assess how well the efficiency of experimental designs can be improved once spatial and genetical correlation is considered. In particular, the evaluation of algorithm efficiency to improve experimental designs focused on the use of RCB designs in field trials with application in plant breeding. Family structure such as half-sib or full-sib families requires appropriate modelling of their genetic relationships (i.e., correlations) and similarly, their physical proximity within rows and/or columns needs to be accounted for as genotypes in close range will share microsite and thus will be correlated. Here, accounting for spatial correlations within rows and columns was necessary to minimize this experimental bias. Incorporation of these correlations not only on the desing stage but also in the analysis stage has been shown by Gezan et al. [5], for RCB desgins, to produce designs that are nearly as efficient as those generated using more complex models such as row-column designs that were analyzed assuming uncorrelated residual errors.

From the detailed practical example that examined a specific condition, results indicated that the SP algorithm is the best as it managed to improve the initial experiment by reducing the average variance of treatment effects by 6.713%. This was followed closely by the SA algorithm with an ODE of 6.258%. The more aggressive algorithm GP4, under the evaluated experimental conditions, was underperforming but it might do better in other design conditions. Also, the algorithm GN had only 12 successful swaps, which was much less than SP and SA algorithms which recorded 192 and 139 swaps, respectively.

Results from Tables 2 and 3 have shown that SP and SA algorithms achieved the highest relative design efficiencies under all experimental conditions for  $\Omega_A^{(30)}$  and  $\Omega_A^{(196)}$  scenarios with the next best algorithm appearing to be the GP4 algorithm followed by GP14 for  $\Omega_A^{(30)}$  scenario or GP98 for  $\Omega_A^{(196)}$  scenario, and last by the GN algorithm. These results could be attributable to the fact that SP swaps a single pair of treatments per iteration, thus taking small steps in the search for an optimal design which makes it more likely to find an optimal condition than GP algorithms that take larger random steps. SA algorithm performed well under  $A$ -optimality criterion since it has the ability not to be trapped in a local minima by accepting a proportion of bad solutions using an exponential distribution and a cooling schedule. It is expected that this algorithm will have better performance in the case of hundreds or thousand entries, where the likelihood of being trapped in local minuma is higher. SA algorithm achieved the lowest relative design efficiencies for the same number of iterations of 5,000 under  $\Omega_D^{(30)}$  scenario as shown in Table 4. It is not very clear why this occurs, but it was observed that it accepted too many bad (or random) solutions as it tried not to be trapped in a local minima, hence, making its progress difficult to maximize the objective function.

This study demonstrated that the incorporation of genetic relationships can affect the optimality of a given design. However, large design improvements have been observed among genetically unrelated individuals, which agrees with findings from Filho and Gilmour [25] although they did not analyse varied levels of spatial correlations. Optimization based on  $A$ -criterion has revealed, from the present study, that a substantial decrease in average variance of treatment effects (i.e., trace) among full-sib families can be achieved for treatments with small levels of narrow-sense heritabilities ( $h^2 = 0.1$ ). For the case of full-sibs with large narrow-sense heritabilities levels such as 0.6 and with a spatial correlation of 0.6, little improvements on the design efficiencies were noted. For experimental designs that were evaluated under  $\Omega_A^{(30)}$  scenario, the amount of design improvement was, for some conditions, about four times larger than that realized under  $\Omega_A^{(196)}$  scenario. This means that more iterations (>50,000) might be required for larger experiments than it would take for a smaller experiment to reach an adequate optimal solution [10]. The number of successful swaps displayed in Figure 3 indicates that they decrease with increasing heritability for all families for experiments evaluated under  $\Omega_A^{(196)}$  and  $\Omega_A^{(30)}$  scenarios for almost all algorithms.

The choice of *A*- or *D*-optimality criteria depends on the desired objective function to be minimized. Both criteria are a convex function of eigenvalues [11,13]. Here, *A*-optimality is a function of the arithmetic mean of the eigenvalues of this matrix whereas *D*-optimality is a function of the geometric mean of these eigenvalues [11]. It is recommended to favor the use of the *A*-optimality criterion, given the additional computational time required to calculate the determinant within the *D*-optimality, particularly for large experiments such as the  $\Omega_A^{(196)}$  scenarios. If additional approximations to the procedures are required to accelerate the optimization process, then a similar approach to the one described by Butler et al. [26] can be implemented.

The algorithms and procedures presented in this study can be easily extended to other complex experimental designs such as non-orthogonal experiments that can be implemented with appropriate extensions of the linear mixed models together with an optimality criterion of choice. In addition, other variants of the search algorithms can also be used; for instance, for the GN algorithm a value different from 0.25 could be chosen to determine when, and which, treatments should be swapped. It was not evaluated if changing this threshold value would increase the efficiency of the GN algorithm.

In summary, the potential to improve experimental designs such as RCB designs has been shown in this study to be highest when SP and SA algorithms were used under *A*-optimality criterion. For both *A*- and *D*-optimality criteria, SP presented the highest overall design efficiencies. In conclusion, the use of a SP algorithm based on *A*-optimality criterion, under a linear mixed model framework that incorporates genetic relatedness and/or spatial correlations is promising. The procedure enables generation of more efficient field designs (by reducing the average variance of the treatment effects) to be used in operational plant breeding programs or in other design of experiments.

**Supplementary Materials:** The following are available online at <http://www.mdpi.com/1999-4893/11/12/212/s1>: Examples.pdf, ped30hs.csv, ped30fs.csv, ped196HS.csv, ped196FS.csv, and final.R.

**Author Contributions:** L.K.M. and S.A.G. conceived and designed the experiments and wrote the paper; L.K.M. performed the experiments and analyzed the data.

**Funding:** This research received no external funding.

**Acknowledgments:** The authors would like to thank the University of Florida's Institute of Food and Agricultural Sciences (UF/IFAS) for funding the study as part of a Ph.D. thesis for Mramba.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A. R-Code for the Algorithms

### Appendix A.1. Simple Pairwise Algorithm (SP)

```
Optimize.rcbd<- function(matdf,n,traceI,criteria,Rinv,Ginv,K) {
  newmatdf <- matdf
  trace <- traceI
  mat <- NULL
  mat <- rbind(mat, c(value = trace, iterations = 0))
  Design_best <- newmatdf
  Des <- list()
  TRACE <- c()
  newmatdf <- SwapPair(matdf = matdf)
  for (i in 2:n) {
    newmatdf <- SwapPair(matdf = newmatdf)
    TRACE[i] <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)
    Des[[i]] <- newmatdf
    if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) < trace) {
      print(sprintf("Swapping within blocks: %d", i, "complete\n",
        sep = ""))
    }
    Design_best <- Des[[i]] <- newmatdf
  }
  Design_best <- newmatdf
}
```

```

trace <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)
mat <- rbind(mat, c(trace = trace, iterations = i))
}
if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) > trace & nrow(mat) <= 1) {
newmatdf <- matdf
Des[[i]] <- matdf
Design_best <- matdf
}
if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) > trace & nrow(mat) > 1) {
newmatdf <- Des[[length(Des) - 1]]
Des[[i]] <- newmatdf
Design_best <- newmatdf
}
}
ODE = (((mat[1,"value"]) - (mat[nrow(mat),"value"]))/(mat[1,"value"]))*100
print(sprintf("ODE due to swapping pairs of treatments within blocks is: %f",
ODE, "complete\n", sep = ""))
list	TRACE = c(as.vector(mat[1, "value"]), TRACE[!is.na(TRACE)]), mat = mat,
Design_best = Design_best)
}

```

## Appendix A.2. Simulated Annealing Algorithm (SA)

```

Optimize_SimAnn_rcbd<- function(matdf,n,traceI,criteria,Rinv,Ginv,K) {
newmatdf <- matdf
trace <- traceI
mat <- NULL
mat <- rbind(mat, c(value = trace, iterations = 0))
Design_best <- newmatdf
Des <- list()
TRACE <- c()
newmatdf <- SwapPair(matdf = matdf)
for (i in 2:n) {
newmatdf <- SwapPair(matdf = newmatdf)
TRACE[i] <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)
Des[[i]] <- newmatdf
if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) < trace) {
print(sprintf("Swapping within blocks: %d", i, "complete\n",
sep = ""))
Design_best <- Des[[i]] <- newmatdf
Design_best <- newmatdf
trace <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)
mat <- rbind(mat, c(trace = trace, iterations = i))
}
Temp<-c()
if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) > trace)
{
dif <- setdiff(NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K),trace)
Temp[i] <- 1/i
accept = exp(-dif/Temp[i])
u = runif(1)
if (u < accept){
Design_best <- Des[[i]] <- newmatdf
Design_best <- newmatdf
trace <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)

```



```

}
if (u > accept & nrow(mat) <= 1) {
  newmatdf <- matdf
  Des[[i]] <- matdf
  Design_best <- matdf
}
if (u > accept & nrow(mat) > 1) {
  newmatdf <- Des[[length(Des) - 1]]
  Des[[i]] <- newmatdf
  Design_best <- newmatdf
}
}
}
}
ODE = (((mat[1,"value"]) - (mat[nrow(mat),"value"]))/(mat[1,"value"]))*100
print(sprintf("ODE due to simulated annealing is: %f", ODE, "complete\n",
  sep = ""))
list(TRACE = c(as.vector(mat[1, "value"]), TRACE[!is.na(TRACE)]), mat = mat,
  Design_best = Design_best)
}

```

### Appendix A.3. Greedy Pairwise Algorithm (GP)

```

OptimizeGreedy.rcbd<- function(matdf,n,traceI,criteria,gsiz,e,Rinv,Ginv,K) {
  newmatdf <- matdf
  trace <- traceI
  mat <- NULL
  mat <- rbind(mat, c(value = trace, iterations = 0))
  Design_best <- newmatdf
  Des <- list()
  TRACE <- c()
  newmatdf <- SwapGreedy(matdf = matdf,gsiz,e = gsiz,e)
  for (i in 2:n) {
    newmatdf <- SwapGreedy(matdf = newmatdf,gsiz,e = gsiz,e)
    TRACE[i] <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)
    Des[[i]] <- newmatdf
    if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) < trace) {
      print(sprintf("Swapping greedily within blocks: %d", i, "complete\n",
        sep = ""))
      Design_best <- Des[[i]] <- newmatdf
      Design_best <- newmatdf
      trace <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)
      mat <- rbind(mat, c(trace = trace, iterations = i))
    }
    if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) > trace & nrow(mat) <= 1) {
      newmatdf <- matdf
      Des[[i]] <- matdf
      Design_best <- matdf
    }
    if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) > trace & nrow(mat) > 1) {
      newmatdf <- Des[[length(Des) - 1]]
      Des[[i]] <- newmatdf
      Design_best <- newmatdf
    }
  }
  ODE = (((mat[1,"value"]) - (mat[nrow(mat),"value"]))/(mat[1,"value"]))*100

```

```

print(sprintf("ODE due to greedily swapping pairs of treatments within
blocks is: %f", ODE, "complete\n", sep = ""))
list(TRACE = c(as.vector(mat[1, "value"]), TRACE[!is.na(TRACE)]), mat = mat,
Design_best = Design_best)
}

```

#### Appendix A.4. Genetic Neighborhood Algorithm (GN)

```

Optimize_GNN_rcbd<- function(matdf,n,traceI,criteria,Amat, Rinv, Ginv, K) {
newmatdf <- matdf
trace <- traceI
mat <- NULL
mat <- rbind(mat, c(value = trace, iterations = 0))
Design_best <- newmatdf
Des <- list()
TRACE <- c()
newmatdf <- Neighbor_rcbd(matdf,Amat)
for (i in 2:n) {
newmatdf <- Neighbor_rcbd(matdf,Amat)
TRACE[i] <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)
Des[[i]] <- newmatdf
if (NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) < trace) {
print(sprintf("Swapping treatments: %d", i, "complete\n",
sep = ""))
Design_best <- Des[[i]] <- newmatdf
Design_best <- newmatdf
trace <- NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K)
mat <- rbind(mat, c(trace = trace, iterations = i))
}
if(NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) > trace && nrow(mat)<=1){
newmatdf <- matdf
Des[[i]] <- matdf
Design_best <- matdf
}
if(NewValue.rcbd(matdf=newmatdf, criteria, Rinv, Ginv, K) > trace && nrow(mat)>1){
newmatdf <- Des[[length(Des) - 1]]
Des[[i]] <- newmatdf
Design_best <- newmatdf
}
}
ODE = (((mat[1,"value"]) - (mat[nrow(mat),"value"]))/(mat[1,"value"]))*100
print(sprintf("ODE due to applying GNN procedure: %f", ODE, "complete\n",
sep = ""))
list(TRACE = c(as.vector(mat[1, "value"]), TRACE[!is.na(TRACE)]), mat = mat,
Design_best = Design_best)
}

```

The GN algorithm also requires the function *Neighbor\_rcbd*:

```

Neighbor_rcbd<-function(matdf,Amat) {
bl<-sample(matdf[, "Reps"],1)
temp1<- matdf[matdf[, "Reps"] == bl,]
rb <-length(unique(temp1[, "Row"]))
cb <-length(unique(temp1[, "Col"]))
mat<-matrix(c(temp1[, "Treatments"]),nrow=rb,ncol=cb,byrow=TRUE)
x<-sample(temp1[, "Treatments"],1)

```

```

m2<-cbind(NA,rbind(NA,mat,NA),NA)
cord <- expand.grid(Row = 1:rb, Col = 1:cb)
ret<-c()
for(i in 1:-1)
for(j in 1:-1)
if(i!=0 || j !=0)
ret<-rbind(ret,m2[cord$Row+i+1+nrow(m2)*(cord$Col+j)])
neigh<-ret[,which(mat==x)]
neigh<-c(x,neigh[!is.na(neigh)]) # add the genotype plus its neighbors
test<-t(combn(neigh,2))
temp<-cbind(test[,1],test[,2],Amat[test])
swapsData<-subset(temp, temp[,3] >= 0.25)
samp1 <- setdiff(as.vector(temp1[, "Treatments"]),c(x,as.vector(temp[,2])))
if(nrow(swapsData)!=0 && length(samp1) !=0)
{
val1<-c()
val2<-c()
for(i in 1:nrow(swapsData))
{
val1[i]<-swapsData[i,1]
val2[i]<-sample(samp1,1)
matdf<-Swap_Specific(matdf,g1=val1[i],g2=val2[i],b1=b1)
}
return(matdf[order(matdf[, "Row"],matdf[, "Col"]),])
}
if(nrow(swapsData)!=0 && length(samp1)==0)
{
val1<-c()
val2<-c()
for(i in 1:nrow(swapsData))
{
val1[i]=swapsData[i,1]
val2[i]=swapsData[i,2]
matdf<-Swap_Specific(matdf,g1=val1[i],g2=val2[i],b1=b1)
}
return(matdf[order(matdf[, "Row"],matdf[, "Col"]),])
}
if(nrow(swapsData)==0) return(SwapPair(matdf))
}

```

## Appendix B. R-Code for Generating Initial Randomized Complete Block Design (RCBD)

### Appendix B.1. Generate a RCBD

```

rcbd<- function(blocks, Treatments,rb,cb, Tr, Tc, irregular=FALSE) {
genot0 <- as.numeric(as.factor(Treatments))
n <- length(Treatments)
Treatments <- c(replicate(blocks, sample(genot0, n, replace = FALSE)))
Reps <- rep(1:blocks, each = length(unique(Treatments)))
if(blocks==1){
cord <- cbind(Row=rep(1:Tr,each=Tc), Col=rep(1:Tc,Tr))
matdf <- cbind(cord, Reps=c(rep(1,n)),Treatments)
row.names(matdf)<-NULL
}
if(cb == Tc & Tr > rb & irregular==FALSE){

```

```

cord <- cbind(Row=rep(1:Tr,each=cb), Col=rep(1:Tc,rb))
matdf <- cbind(cord,Reps, Treatments)
row.names(matdf)<-NULL
}
if(rb == Tr & Tc > cb & irregular==FALSE){
Row=rep(rep(1:Tr,each=cb),Tc/cb)
Col <- rep(split(1:Tc, cut(seq_along(1:Tc), blocks, labels = FALSE)),each=Tr)
Col <- unlist(Col)
cord <- cbind(Row,Col)
matdf <- cbind(cord,Reps, Treatments)
row.names(matdf)<-NULL
}
if(Tr > rb & Tc > cb & irregular==FALSE){
Row=rep(rep(1:Tr,each=cb),Tc/cb)
Col <- rep(split(1:Tc, cut(seq_along(1:Tc), Tc/cb, labels = FALSE)),each=Tr)
Col <- unlist(Col)
cord <- cbind(Row,Col)
matdf <- cbind(cord,Reps, Treatments)
row.names(matdf)<-NULL
}
if(irregular==TRUE){
cord <-cbind(Row=Row,Col=Col)
matdf <- cbind(cord, Reps, Treatments)
row.names(matdf)<-NULL
}
matdf[order(matdf[, "Row"],matdf[, "Col"]),]
}

```

### Appendix B.2. Generate Multiple RCBD

```

MultipleDesigns <- function(DesN, blocks, Treatments, rb,cb, Tr, Tc,Amat=FALSE,
criteria="A", h2, rhox,rhoy, s20,irregular=FALSE) {
matrix0 <- list()
initialValues1 <- c()
initialValues2 <- c()
for (i in 1:DesN) {
print(sprintf("generating initial design: %d", i, "complete\n",
sep = ""))
flush.console()
matrix0[[i]] <- rcbd(blocks, Treatments, rb,cb, Tr, Tc, irregular)
initialValues1[i] <- VarCov.rcbd(matdf = matrix0[[i]], rhox,rhoy,
h2, s20, Tr, Tc, criteria, Amat, irregular)[[1]]
a <- which.min(initialValues1)
newmatdfA <- matrix0[a][[1]]
min_initialValues1 <- initialValues1[a][[1]]
}
return(list(newmatdf = newmatdfA, trace0 = initialValues1,
min_value = min_initialValues1, meanA = mean(initialValues1)))
}

```

### Appendix C. Generate a Numerator Relationship Matrix

```

GenA <- function(male, female) {
if (nargs() == 1) {

```

```

stop("require male and female entries")
}
if (length(male) != length(female)) {
  stop("length of male and female differ")
}
male[is.na(male)] <- 0 # convert all NA to zeros
female[is.na(female)] <- 0 # convert all NA to zeros
n <- length(male)
N <- n + 1
A <- matrix(0, ncol = N, nrow = N)
male <- (male == 0) * (N) + male
female <- (female == 0) * N + female
for (i in 1:n) {
  A[i, i] <- 1 + A[male[i], female[i]]/2
  for (j in (i + 1):n) {
    if (j > n)
      break
    A[i, j] <- (A[i, male[j]] + A[i, female[j]])/2
    A[j, i] <- A[i, j]
  }
}
A <- as(A, "sparseMatrix")
return(A[1:n, 1:n])
}

```

## Appendix D. Calculate the Variance-Covariance Matrix

```

VarCov.rcbd <- function(matdf, rhox, rhoxy, h2, s20, Tr, Tc, criteria="A",
  Amat=FALSE,irregular=FALSE) {
  if(nrow(matdf)==length(unique(matdf[, "Treatments"]))) {
    X <- as.matrix(matdf[, "Reps"])
  }
  if(nrow(matdf) > length(unique(matdf[, "Treatments"]))) {
    X <- Matrix::sparse.model.matrix(~as.factor(matdf[, "Reps"])-1)
  }
  s2e <- (1 - s20) * (1 - h2)
  stopifnot(s2e > 0)
  m = length(unique(matdf[, "Treatments"]))
  if(is.matrix(Amat)){
    G <- h2 * as.matrix(Amat)
    Ginv <- round(chol2inv(chol(as.matrix(G))),7)
    Ginv <- as(Ginv, "sparseMatrix")
  }
  else{
    Ginv <- round((1/h2) * Matrix::Diagonal(m),7)
    Ginv <- as(Ginv, "sparseMatrix")
  }
  Z<- Matrix::sparse.model.matrix(~as.factor(matdf[, "Treatments"]) - 1)
  # calculating R and its inverse for spatial analysis
  bb <- length(unique(matdf[, "Reps"]))
  matdf <- matdf[order(matdf[, "Row"],matdf[, "Col"]),]
  if(irregular==TRUE){
    R <- Matrix::Diagonal(nrow(matdf))
    for(i in 1:(nrow(matdf)-1)) {
      x1 <- matdf[, "Col"][i]

```

```

y1 <- matdf[, "Row"][i]
for (j in (i+1):nrow(matdf)){
  x2 <- matdf[, "Col"][j]
  y2 <- matdf[, "Row"][j]
  R[i,j]<-(rhox^abs(x2 -x1))*(rhoy^abs(y2 -y1))
}
}
R = as.matrix(round(s2e*R,7))
R[lower.tri(R)] <- t(R)[lower.tri(R)]
R <- as(R, "sparseMatrix")
Rinv <- round(chol2inv(chol(R)),7)
Rinv <- as(Rinv, "sparseMatrix")
}
if(irregular==FALSE){
  sigx <- Matrix::Diagonal(Tc)
  sigx <- rhox^abs(row(sigx) - col(sigx))
  sigy <- Matrix::Diagonal(Tr)
  sigy <- rhoy^abs(row(sigy) - col(sigy))
  R <- round(s2e * kronecker(sigy, sigx),7)
  R <- as(R, "sparseMatrix")
  Rinv <- round(chol2inv(chol(R)),7)
  Rinv <- as(Rinv, "sparseMatrix")
}
C11 <- Matrix::crossprod(as.matrix(X), as.matrix(Rinv)) %*% as.matrix(X)
C11inv <- solve(C11)
k1 <- Rinv %*% as.matrix(X)
k2 <- Matrix::tcrossprod(as.matrix(C11inv), as.matrix(X))
k3 <- k2 %*% Rinv
K <- k1 %*% k3
K <- as(K, "sparseMatrix")
temp0 <- Matrix::crossprod(Z, Rinv) %*% Z + Ginv - Matrix::crossprod(Z, K) %*% Z
C22 <- solve(temp0)
C22 <- as(C22, "sparseMatrix")
Ginv = round(Ginv,7)
Rinv = Matrix::drop0(round(Rinv,7))
K = round(K,7)
C22 = round(C22,7)
if (criteria == "A") {
  return(c(traceI = sum(Matrix::diag(C22)), Ginv = Ginv, Rinv = Rinv, K=K))
}
if (criteria == "D") {
  deTm = Matrix::det(C22)
  return(c(doptimI = log(deTm), Ginv = Ginv, Rinv = Rinv, K=K))
}
}
}

```

## References

1. Welham, S.J.; Gezan, S.A.; Clark, S.J.; Mead, A. *Statistical Methods in Biology; Design and Analysis of Experiments and Regression*; Chapman & Hall: Boca Raton, FL, USA, 2015.
2. Piepho, H.P.; Möhring, J.; Melchinger, A.E.; Büchse, A. BLUP for phenotypic selection in plant breeding and variety testing. *Euphytica* **2008**, *161*, 209–228. [[CrossRef](#)]
3. John, J.A.; Williams, E.R. *Cyclic and Computer Generated Designs*, 2nd ed.; Monographs of Statistics and Applied Probability 38; Chapman and Hall: London, UK, 1995.



4. Williams, E.R.; John, J.A.; Whitaker, D. Construction of resolvable spatial row-column designs. *Biometrics* **2006**, *62*, 103–108. [CrossRef] [PubMed]
5. Gezan, S.A.; White, T.L.; Huber, D.A. Accounting for spatial variability in breeding trials: A simulation study. *Agronomy* **2010**, *102*, 1562–1571. [CrossRef]
6. Butler, D.G.; Smith, A.B.; Cullis, B.R. On the design of field experiments with correlated treatment effects. *J. Agric. Biol. Environ. Stat.* **2014**, *19*, 539–555. [CrossRef]
7. Chernoff, H. Locally optimal designs for estimating parameters. *Ann. Math. Stat.* **1953**, *24*, 586–602. [CrossRef]
8. Cullis, B.R.; Lill, W.; Fisher, J.; Read, B.; Gleeson, A. A new procedure for the analysis of early generation variety trials. *J. R. Stat. Soc. Ser. C (Appl. Stat.)* **1989**, *38*, 361–375. [CrossRef]
9. Cullis, B.R.; Smith, A.B.; Coombes, N.E. On the design of early generation variety trials with correlated data. *J. Agric. Biol. Environ. Stat.* **2006**, *11*, 381–393. [CrossRef]
10. Mramba, L.K.; Peter, G.F.; Whitaker, V.M.; Gezan, S.A. Generating improved experimental designs with spatially and genetically correlated observations using mixed models. *Agronomy* **2018**, *8*, 40. [CrossRef]
11. Kuhfeld, W.F. *MR-2010C—Experimental Design: Efficiency, Coding, and Choice Designs*; Technical Report; SAS Institute Inc.: Cary, NC, USA, 2010.
12. Wald, A. On the efficient design of statistical investigations. *Ann. Math. Stat.* **1943**, *14*, 134–140. [CrossRef]
13. Das, A. An introduction to optimality criteria and some results on optimal block design. In *Design Workshop Lecture Notes*; Indian Statistical Institute: Kolkata, India; Theoretical Statistics and Mathematics Unit: New Delhi, India, 2002; pp. 1–21.
14. Kirkpatrick, S.; Gelatt, C.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
15. VSN International. *CycDesign 6.0: A Package for the Computer Generation of Experimental Designs*; VSN International Ltd.: Hemel Hempstead, UK, 2018.
16. VSN International. *Genstat for Windows*, 19th ed.; VSN International Ltd.: Hemel Hempstead, UK, 2017.
17. Coombes, N.E. DiGger: Design Search Tool in R. 2009. Available online: <http://nswdpibiom.org/austatgen/software/> (accessed on 17 December 2018).
18. Cressie, N.A.C. *Statistics for Spatial Data*, revised ed.; John Wiley & Sons, Inc.: New York, NY, USA, 1993.
19. Gilmour, A.R.; Gogel, B.J.; Cullis, B.R.; Thompson, R. *ASReml User Guide Release 3.0*; VSN International Ltd.: Hemel Hempstead, UK, 2009.
20. Henderson, C.R. The estimation of genetic parameters. *Ann. Math. Stat.* **1950**, *21*, 309–310.
21. Hooks, T.; Marx, D.; Kachman, S.; Pedersen, J. Optimality criteria for models with random effects. *Revista Colombiana de Estadística* **2009**, *32*, 17–31.
22. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2018.
23. Borges, P.; Eid, T.; Bergseng, E. Applying simulated annealing using different methods for the neighborhood search in forest planning problems. *Eur. J. Oper. Res.* **2014**, *233*, 700–710. [CrossRef]
24. Liu, G.; Han, S.; Zhao, X.; Nelson, J.D.; Wang, H.; Wang, W. Optimisation algorithms for spatially constrained forest planning. *Ecol. Model.* **2006**, *194*, 421–428. [CrossRef]
25. Filho, J.S.B.; Gilmour, S.G. Planning incomplete block experiments when treatments are genetically related. *Biometrics* **2003**, *59*, 375–381. [CrossRef]
26. Butler, D.G.; Eccleston, J.A.; Cullis, B.R. On an approximate optimality criterion for the design of field experiments under spatial dependence. *Aust. N. Z. J. Stat.* **2008**, *50*, 295–307. [CrossRef]

