Research Article

Applying the Median and Genetic Algorithm to Construct *D*- and *G*-optimal Robust Designs Against Missing Data

Sitisak Mahachaichanakul and Patchanok Srisuradetchai*
Department of Mathematics and Statistics, Faculty of Science and Technology, Thammasat University, Rangsit Campus, Pathum-Thani, Thailand

* Corresponding author. E-mail: spatchan@tu.ac.th DOI: 10.14416/j.ijast.2018.10.002
Received: 23 May 2017; Revised: 23 August 2017; Accepted: 28 August 2017; Published online: 4 October 2018

Abstract
In practice, there is a circumstance in which some observed values in well-planned experiments are missing. In this research, small optimal robust response surface designs against missing data were constructed using a Genetic Algorithm (GA) with a Minimum (Min) of alphabetic criteria such as *D*- and *G*-optimality for a second-order model. The resulting designs from GA were compared to designs generated from Exchange Algorithm (EA). Unlike EA, GA uses a set of continuous design points as candidate points, so GA produces more optimal and robust designs. For *D*-optimality, the results showed that the values for *D*-efficiency, Min *D*, Med *D*, and leave-1-out *D* criteria of designs generated by GA were all greater than or equal to those from EA. Calculated by EA and GA methods, all *G*-related criteria values were less than 0.6 apart, except in the case of $N = 7$. Furthermore, a median of alphabetic optimality criteria has been proposed for use as a criterion to construct robust designs. This approach compromises between optimality criteria such as usual *D*- and *G*-optimality and pessimistic-oriented criteria such as Min *D*- and Min *G*-optimality. For general missing points, the Med *D*-optimal designs would be superior to the Min *D*-optimal designs, especially for very small designs. The Med *G*-optimal designs are far better than the *G*-optimal designs, although the sample size is increased.

Keywords: Experimental designs, Response surface designs, Genetic algorithm, Missing data

1 Introduction

A response surface design is an experimental design for developing, improving, and optimizing products. It has been used in many industrial researches. The design contains factor settings that either maximize or minimize single or multiple responses. A small exact response surface design is usually constructed using a second-order polynomial model, as shown in Equation (1).

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{j=i+1}^{k} \sum_{i=1}^{k-1} \beta_{ij} x_i x_j + \varepsilon \qquad (1)$$

where y is the measured response; $x_1, x_2, ..., x_k$ are the design variables or predictors; β's are the parameter coefficients and ε is a random error term. The second-order model is the most popular model used to approximate the true relationship between the response and predictors. This is because the fitted surface of the model has a simple interpretation with respect to the maximum, minimum, and saddle points [1].

The matrix notation of the second-order model is in Equation (2).

$$y = \beta_0 + \mathbf{x}^{\mathrm{T}}\mathbf{b} + \mathbf{x}^{\mathrm{T}}\mathbf{B}\mathbf{x} + \varepsilon = \mathbf{X}\beta + \varepsilon, \qquad (2)$$

where $\mathbf{X}$ is called a "model matrix":

Please cite this article as: S. Mahachaichanakul and P. Srisuradetchai, "Applying the median and genetic algorithm to construct *D*- and *G*-optimality robust designs against missing data," *Applied Science and Engineering Progress*, vol. 12, no. 1, pp. 3–13, Jan.–Mar. 2019.

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11}\cdots x_{1k} & x_{11}x_{12}\cdots x_{1(k-1)}x_{1k} & x_{11}^2\cdots x_{1k}^2 \\ 1 & x_{21}\cdots x_{2k} & x_{21}x_{22}\cdots x_{2(k-1)}x_{2k} & x_{21}^2\cdots x_{2k}^2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N1}\cdots x_{Nk} & x_{N1}x_{N2}\cdots x_{N(k-1)}x_{1k} & x_{N1}^2\cdots x_{Nk}^2 \end{bmatrix}.$$

Borkowski [2] used design optimality criteria such as *D* and *G* criteria with GA to construct and evaluate the exact *N*-point design. An example of the resulting 7-point *D*-optimal is shown in Figure 1. An exact design is one that has the number of design points as a specified integer *N*. The exact *N*-point design can be represented by an $N\times k$ design matrix, where each of the *N* rows corresponds to the experimental point and has *k* columns or *k* factors. Therefore, the design is expanded into an $N\times p$ expanded design matrix ***X***, which is used to calculate the design optimality criteria of interest. Corresponding to Equation (1), the number of model parameters is the number of columns in matrix **X**, $p = \left(\frac{k+2}{2}\right)$.

The design optimality criteria, also called alphabetic optimality criteria, can be divided into 2 categories. The first category focuses on minimizing the generalized variance of parameter estimates such as *D*-optimality, introduced by Wald [3] in 1943. The *D*-optimality criterion minimizes the determinant $|(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}|$ and is directly proportional to the generalized variance. This is equivalent to maximizing $|\mathbf{X}^{\mathrm{T}}\mathbf{X}|$, a part of the Fisher information for exact designs [1]. *D* stands for "determinant". If ξ^* is a *D*-optimal design, ξ^* satisfies Equation (3) below.

$$\xi^* = \arg\min_{\xi\in\Xi}\left|\mathbf{M}^{-1}(\xi)\right| = \arg\min_{\xi\in\Xi}\left|N(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\right|, \qquad (3)$$

where Ξ is a set of all possible continuous designs on design space χ and $\mathbf{M} = \mathbf{X}^{\mathrm{T}}\mathbf{X}/N$ is a moment matrix [4]. To compare *D*-optimal robust designs, *D*-efficiency as defined in Equation (4) is usually used as a criterion. Higher *D*-efficiency in a design means the design is better.

$$D\text{-efficiency} = \left(\frac{\left|\mathbf{X}^{\mathrm{T}}\mathbf{X}\right|}{N^p}\right)^{1/p}\times 100 = \frac{\left|\mathbf{X}^{\mathrm{T}}\mathbf{X}\right|^{1/p}}{N}\times 100 \qquad (4)$$

The second category of design optimality criteria centers on minimizing the prediction variance. In this paper, *G*-optimality is chosen as it is generally used in literature. *G*-optimality, introduced by Smith [5], minimizes the maximum of the Scaled Prediction Variances (SPVs). Since the SPV is calculated by all points in design space χ, *G* stands for "global." If design ξ^* is a *G*-optimal, ξ^* satisfies the condition in Equation (5).

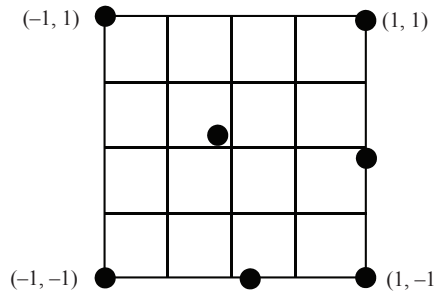

Figure 1: Example of 7-point *D*-optimal exact design for a second-order model in two factors.

$$\begin{aligned}\xi^* &= \arg\min_{\xi\in\Xi}\max_{\mathbf{x}\in\chi}\mathrm{SPV}(\mathbf{x}) \\ &= \arg\min_{\xi\in\Xi}\max_{\mathbf{x}\in\chi} N\mathbf{x}^{(m)^{\mathrm{T}}}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{x}^{(m)}, \end{aligned} \qquad (5)$$

where $\mathbf{x}^{(m)} = [1, x_1, \ldots, x_k, x_1^2, \ldots, x_k^2, x_1x_2, \ldots, x_{k-1}x_k]$. Likewise, the *G*-efficiency shown in Equation (6) is used to compare resulting *G*-optimal robust designs.

$$G\text{-efficiency} = \frac{p}{\max\limits_{\mathrm{x}\in\chi}\mathrm{SPV(x)}}\times 100 \qquad (6)$$

In practice, the data in well-planned experimentations has the potential to be missing or lost. The designs from Borkowski [2] can be considered optimal designs when there is no missing data. However, there are some *D*-optimal designs with only one missing design point, but the corresponding $|\mathbf{X}^{\mathrm{T}}\mathbf{X}|$ can be 0. This causes an inability to estimate parameters, as matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is singular [6]. For example, $|\mathbf{X}^{\mathrm{T}}\mathbf{X}|$ will be zero if point (–1, –1) is missing, as seen in Figure 1.

Srisuradetchai [6] introduced robust response surface designs against missing data for alphabetic optimal designs by introducing the minimum of interested efficiencies calculated on an $(N-1)$-point design. The idea behind this is that, if one design point is missing, design efficiency will be calculated

from the remaining $N-1$ design points. Applying the aforementioned approach to every possible missing point in the design, there will be N D-efficiencies and the minimum will be used as modified criteria to construct a robust design. This modified criterion is considered a pessimistic approach because the minimum is used.

Let Ξ be a set of all possible exact designs on design space χ, Θ be a set of all design points in the design, and $\mathbf{x}_i$ be a design point in Θ. D-optimal robust exact design ξ^* satisfies

$$\xi^* = \arg \max_{\xi \in \Xi} \min_{\mathbf{x}_i \in \Theta} \left| \mathbf{M}(\xi_{-\mathbf{x}_i}) \right|, \quad (7)$$

where $\mathbf{M}(\xi_{-\mathbf{x}_i})$ is a moment matrix of design ξ without design point $\mathbf{x}_i$. The resulting design constructed from Min D-optimal criterion assures that the minimum of D-efficiencies is still highest. Using this method will prevent the worst-case scenario, where a missing point leads to the minimum D-efficiency. The criterion will be called "Min D". For Min G-optimality criterion, let χ be the design space and $\mathbf{a} \in x$. Srisuradetchai [6] seeks a design ξ^* satisfying

$$\xi^* = \arg \min_{\xi \in \Xi} \max_{\mathbf{x}_i \in \Theta} [\max_{\mathbf{a} \in \chi} \mathbf{a}^{(m)^{\mathrm{T}}} \mathbf{M}^{-1}(\xi_{-\mathbf{x}_i}) \mathbf{a}^{(m)}], \quad (8)$$

where $\mathbf{M}^{-1}(\xi_{-\mathbf{x}_i})$ is an inverse of the moment matrix of design ξ without design point $\mathbf{x}_i$. The resulting design achieving Equation (8) is called Min G-optimal robust exact design.

The method used to construct robust response surface designs against missing data in Srisuradetchai [6] is EA, where the process is started from constructing an initial design and setting candidate points for exchange. A design point in the initial design will be sequentially chosen from the first to the last. Each of the initial design points will be sequentially exchanged with all candidate points. A design efficiency of interest will be calculated for each exchange. The design with the highest efficiency will be kept and used as the new starting design. The entire process repeats until improvement in the efficiency is not significant. The final design will be kept as the optimal robust exact design.

For response surface designs, independent values will typically be coded to 1 for the highest value and –1 for the lowest value. This is called orthogonal coding. If a one-decimal digit is used, the candidate design points will be –1,–0.9,...,0,...,0.9,1 and the total number of design points is 21 in one-factor designs. For two-factor designs, the total number of design points goes up to 21^2 or 441 and up to 9,261 points for three-factor designs. The number of candidate points becomes a large number, although the number of decimal digits only increases from one to two.

The robust designs constructed using EA from Srisuradetchai [6] have design points with only one decimal digit. It is believed that, as the number of decimal digits increases, design efficiency will increase. This agrees with the actual idea of response surface methodology, which seeks factor settings over a continuous region in a design space. Also, using EA to construct response surface designs is time-consuming, especially when k is greater than 2. For example, our experience was that the design for $N = 7$, $k = 2$, and $\chi = \{-1,-0.9,...,0.9,1\}^2$ took 5– 6 minutes to construct the D-optimal robust design. For $N = 8$, it took about 6–7 minutes. However, the design with $N = 11$, $k = 3$, and $\chi = \{-1,-0.9,...,0.9,1\}^3$ involved the length of time consumed for constructing a D-optimal robust design jumping to 9 hours as the number of candidate points increased exponentially. Srisuradetchai [6] also mentioned that "Probably, using a grid search with the point-exchanging algorithm is not a very good method to find the best Min G design".

In previous literature, Borkowski [2] used GA to construct optimal D, A, G and IV small exact response surface designs for second-order models with 1, 2 and 3 factors assuming that there was no missing data. As GA has never been used to generate robust exact response surface designs against missing points, it is a challenge to see if Min D and Min G values can be improved by GA; this will be our first objective. Also, a custom-tailored genetic algorithm to construct D- and G-optimal robust exact designs against missing data will be described in detail. Finally, robust designs will be generated using the newly-proposed criterion which is a median of $N-1$ design efficiencies.

The scope of the study involves the construction of small optimal robust exact response surface designs against a missing point for a second-order model with 2 factors and 7, 8, 9 and 10 design points. The resulting robust designs will be compared with optimal designs from Borkowski [2] and robust designs from Srisuradetchai [6].

2 Using Median to Construct Robust Designs

The goal of using a minimum of alphabetic efficiencies in Srisuradetchai [6] is to construct robust designs with the highest minimum design efficiency in order to prevent the worst-case scenario. However, the resulting design is generally not near-optimal if there is no missing data. To compromise between the optimality criteria (for example, *D*- and *G*-optimality) and pessimistic-oriented criteria (for example Min *D* and Min *G*), the median will be used in our study instead of the minimum. In other words, we consider that preventing the worst-case scenario is not a priority goal. Instead, we are interested in a design that is robust to a missing point and still near-optimal when there is no missing data. Another reason convincing us to use the median is that our experience has shown that, in a process of constructing robust designs satisfying Equation (8), there will be N design efficiencies. Each of them is calculated from $N-1$ design points. Usually, one or more efficiency values are extreme, so using the median would be reasonable.

Let Ξ denote a set of all possible exact designs on design space χ, Θ be a set of all design points in the design space, and $\mathbf{x}_i$ be a design point in Θ. Thus, the median *D*-optimal or Med *D* design ξ^* is a design satisfying Equation (9).

$$\xi^* = \arg \max_{\xi \in \Xi} \operatorname*{med}_{\mathbf{x}_i \in \Theta} \left| \mathbf{M}(\xi_{-\mathbf{x}_i}) \right| \tag{9}$$

where $\mathbf{M}(\xi_{-\mathbf{x}_i})$ is a moment matrix of design ξ without design point $\mathbf{x}_i$. The resulting design is called Med *D*-optimal robust exact design.

For a median *G*-optimal design, the optimal design ξ^* needs to achieve

$$\xi^* = \arg \min_{\xi \in \Xi} \operatorname*{med}_{\mathbf{x}_i \in \Theta} [\max_{\mathbf{a} \in \chi} \mathbf{a}^{(m)^{\mathrm{T}}} \mathbf{M}^{-1}(\xi_{-\mathbf{x}_i}) \mathbf{a}^{(m)}] \tag{10}$$

The resulting design obtained from Equation (10) is called a Med *G*-optimal robust exact design. In our study, designs satisfying Equations (9) and (10) will be constructed by GA.

3 Genetic Algorithms

Introduced by Holland [7] in 1975, a Genetic Algorithm (GA) is a computer-based strategy for seeking and developing solutions to a problem. The main idea is taken from biological population genetics and natural selection theory. A candidate solution is called a chromosome, as represented by a string of genes that are encoded in binary representations whose values are 0 and 1. In practice, Davis [8] found that GA using real number representations for genes was better than GA with binary representations followed by decoding in numerical optimization problems.

In this research, genes are factor settings that will be optimized after running a certain number of generations of GA with the goal of finding a design that maximizes an objective function. In this case, the objective functions are robust alphabetic criteria such as Min *D*, Min *G*, Med *D*, and Med *G*. For one generation of GA, it contains 4 major steps: initiation, selection, reproduction, and convergence checking. The descriptions for these steps are as follows.

3.1 *Initiation*

The initial population chromosome M is created at the beginning of GA; M is an odd or even number based on the number of the best chromosomes. If we decide to keep just the best chromosomes, M will be an odd number. If we decide to keep a pair of the best chromosomes, M will be an even number. These conditions can be considered in reverse.

A chromosome is represented as an $N \times p$ matrix $\mathbf{X}$ and is also called a "model matrix". Each gene x_{ij} is the coded setting of the ith factor in the jth experimental run. Its value is dictated by the design space χ. In this study, each x_{ij} is in $[-1, 1]$ and M is chosen to be an even number, so a pair of the best chromosomes will be kept in the selection step.

3.2 *Selection*

This step is divided into two differently purposed steps. The first step is to select the best chromosomes from each generation. The best chromosomes have to maximize objective functions: Equations (7), (8), (9), or (10). The number of chromosomes selected is based on the number of the initial population chromosomes M chosen in initiation step. The selected chromosomes are called "elite" chromosomes and will not be changed in the next step of GA, i.e. the reproduction step. The purpose of this step is not only to pass good traits of elite chromosomes to future

generations, but also to prevent inferior chromosomes from reproducing. As a result, the initial population M is reduced to M–2 "parent" chromosomes. The second step is to randomly select a pair of chromosomes from the total (M–2)/2 pairs of parent chromosomes. The selected pair (considered parents) will reproduce 2 new chromosomes called offspring chromosomes. The comparison occurs for each parent-offspring pair, and 2 chromosomes with higher objective function values will be kept for the next generation of GA. If any parent pair of chromosomes is selected, it will not be used again for reproducing offspring. This process repeats until all (M–2)/2 pairs of parent chromosomes reproduce completely.

3.3 *Reproduction*

The reproduction step can be varied according to the researcher and the nature of the solution of interest, but has the same idea as biological population genetics. This step contains two different types of operators: between-chromosome crossover operators and chromosome mutation operators.

To decide whether genes (x_{ij}) in a certain chromosome (**X**) will be changed or not, it depends on PTIP (Probability Test Is Passed) for that chromosome. This is the probability that $0 \le u \le \alpha$ where α is the probability of success in Bernoulli trial and u is a random deviate from uniform distribution on [0,1]. If $0 \le u \le \alpha$, reproduction operators will apply to the corresponding gene or row of the gene in the chromosome matrix based on the type of operators. Otherwise, the gene is left unchanged.

In this study, the probability test of crossover operators is set to $\alpha = 0.2$ for the first half of generations of chromosomes and set to $\alpha = 0.05$ for the second half of generations. In the mutation operators, α is 0.1 for the first quarter of generations of chromosomes and is 0.01 for later generations. The value of α is usually a small number to mimic a gradual change in the natural selection theory.

The chromosomes that go through the defined set of operators – comprising both crossover and mutation operators – are called offspring chromosomes. The crossover operators here are blending and swap. For mutation operators, there are sign change, zero genes, half genes, extreme genes, and creep; these are all used for creating offspring chromosomes. Note that for crossover operators, PTIP will be checked for each row of genes by assigning random deviate u to each row of the matrix. On the other hand, the mutation operators will check PTIP for each gene or element x_{ij} by assigning random deviate u to each element of matrix **X**.

Let matrices **A** and **B** represent two parent chromosomes and matrix $\mathbf{A}^*$ and $\mathbf{B}^*$ represent their offspring chromosomes. The details for each operator are described as follows.

3.3.1 Blending

This operator is a crossover operator that combines the information from rows of genes in **A** and **B** to create a new row of genes injected to the offspring chromosomes. Let A_a be the ath row of **A** and B_b be the bth row of **B**. Blending occurs for the ath row selected by the PTIP condition and the bth row, which is randomly selected by a uniform deviate. A change for each pair of A_a and B_b can be shown in the following linear combination [Equations (11) and (12)]:

$$A_a^* = \delta A_a + (1-\delta) B_b \tag{11}$$

where A_a^* is the ath row in $\mathbf{A}^*$ and

$$B_b^* = \delta B_b + (1-\delta) A_a \tag{12}$$

where B_b^* is the bth row in $\mathbf{B}^*$ The δ is a blending ratio that is a random deviate from uniform distribution [0, 1]. The process is applied from the first PTIP row in **A** through the last PTIP row. For B_b to have a chance to be repeatedly selected, if this occurs, the latest PTIP row in **A** will be the only row chosen to blend with B_b.

3.3.2 Swap

The swap operator, considered a crossover operator, exchanges information from rows of genes in **A** and **B**. Unlike a blending operator, this operator does not change values of rows in the parent chromosomes. Instead, it simply exchanges PTIP rows in parent chromosomes to reproduce offspring chromosomes.

3.3.3 Sign Change

This operator is a mutation operator that changes the sign of gene x_{ij} for which a probability test is passed.

If one PTIP gene is positive, it will be changed to a negative value counterpart. Likewise, a negative value for a PTIP gene will be changed to positive.

3.3.4 Zero Genes

Zero Genes is a mutation operator that changes the value of PTIP gene x_{ij} to zero. Because a design point of response surface designs is coded between –1 and 1, the middle value is zero. The idea behind this kind of operator is to get the middle point 0.

3.3.5 Half Genes

This mutation operator changes the value of PTIP gene x_{ij} to its halved value. For example, if PTIP x_{ij} is 0.25, mutated x_{ij} will be 0.25/2 = 0.125. The idea behind this operator is from Srisuradetchai [6] because resulting robust designs generated from EA have design point values between –0.5 and 0.5. They are middle values between zero and the lowest and highest values, respectively. In practice, exact values –0.5 and 0.5 cannot be obtained through the aforementioned operators, so this kind of operator will be used accordingly.

3.3.6 Extreme Genes

The mutation operator that changes the value of PTIP gene x_{ij} to either –1 or 1 is called Extreme Genes. It can be assumed that robust designs against missing data might have design points containing –1 or 1, extreme values.

3.3.7 Creep

The creep operator will change the value of PTIP gene x_{ij} by adding a very small number. The idea behind this is to slowly change the values in each generation of GA to simulate an actual mutation in real life. This is adapted from the theory of evolution, stating that a mutation must take time to be observed. In this operator, PTIP gene will be added with a random deviate from $N(0,\sigma^2)$, where $0.001 \leq \sigma \leq 0.1$. This also allows GA to produce a robust design in a continuous space.

3.4 *Convergence checking*

After obtaining the total of M–2 offspring chromosomes from the reproduction step, they will be compared with their parent chromosomes. The one with higher design efficiency of interest will be kept and used in the next generation of GA, while lower efficiency chromosomes will be discarded. In total, there will be M–2 resulting chromosomes from the reproduction step and 2 elite chromosomes, selected beforehand in the selection step. Finally, there are M chromosomes in total to be used in the next generation of GA.

For our study, three conditions are used as criteria to stop the repeating process of GA. The first condition is the maximum number of generations in GA. The total number of generations will affect objective function values and the length of time for design construction. Second is the maximum number of times for having the same best designs, one of 2 elite chromosomes. This part is specified because there is a chance that adjacent generations of GA will produce the same design efficiency or the same design because the PTIP condition might not pass at all in a certain generation. The last condition is the minimum change of design efficiencies between the last and current generations. It has to be specified in order to stop the process at a current generation.

In the convergence checking step, a total of 18,000 generations is used with the best design efficiencies being allowed to be the same for 6,000 generations. Also, the minimum change in the last condition is set as 10^{-4}.

4 Results

The comparisons are made in 2 dimensions: algorithms and criteria.

4.1 *Comparing the algorithms*

The resulting N-point Min D-optimal robust designs using EA and GA with their properties are shown in Table 1. It is noticeable that there are two robust designs in which both algorithms produce the same design points: (±1, ±1), (±1,0), and (0,±1) for the 8-point design and (±1, ±1), (±1,0), (0,±1), (0,0) for the 9-point design. For 7- and 10-point designs, Min D-efficiency is higher by using GA. Although it is a small improvement in Min D-efficiencies, this cannot be achieved by EA with a design space grid of 4 decimal digits because of the huge number of candidate points.

Table 1: Design points of *N*-point Min *D*-optimal robust designs using EA and GA and their properties

N	Algorithm	*D*-efficiency	Min *D*	Med *D*	Leave-one-out (Mean *D*)	Design Points
7	Exchange	40.5674	31.5756	31.5756	33.2396	(±1, –1), (±1, 0.7), (0, ±1), (0, 0)*
	Genetic	40.9343	31.6883	31.7902	33.7616	(±1, –1), (–1, 0.1924), (–0.4666, 1), (–0.1775, –0.5799), (1, –0.2747), (1, 1)
8	Exchange	45.4280	38.5145	40.8727	40.8727	(±1, ±1), (±1, 0), (0, ±1)*
	Genetic	45.4280	38.5145	40.8727	40.8727	(±1, ±1), (±1, 0), (0, ±1)
9	Exchange	46.2241	39.5810	45.4280	42.8293	(±1, ±1), (±1, 0), (0, ±1), (0, 0)*
	Genetic	46.2241	39.5810	45.4280	42.8293	(±1, ±1), (±1, 0), (0, ±1), (0, 0)
10	Exchange	44.5166	40.4043	41.9657	42.3082	(±1, ±1), (–1, –0.5), (–1, 1), (0, 0.1), (0.3, –1), (0.5, 1), (1, –0.2)*
	Genetic	44.7694	40.4664	42.2662	42.5406	(±1, ±1), (–1, –0.4974), (–1, 1), (–0.1099, 0.0357), (0.2490, –1), (0.4579, 1), (1, –0.1881)

* Design generated using the Exchange Algorithm (EA) produced by Srisuradetchai [6]

Table 2: Properties of the *N*-point Min *G*-optimal robust designs using EA and GA

N	Algorithm	*G*–efficiency	Min *G*	Med *G*	Leave–one–out (Mean *G*)	Design Points
7	Exchange	37.4269	9.3632	10.0782	14.9361	(±1, –1), (–1, 1), (–0.6, 0.2), (–0.3, –1), (0.2, 1), (1, 0.4)*
	Genetic	35.0715	9.9635	10.1688	14.2102	(–1,–0.9949), (–1, 0.995), (–0.5584, 0.3226), (–0.2848, –1), (0.1984, 1), (0.9692, –1), (1, 0.3863)
8	Exchange	47.8846	18.3553	25.9948	25.9948	(±1, ±1), (–1, –0.3), (–0.3, 1), (0.3, –1), (1, 0.3)*
	Genetic	47.8695	18.3569	26.0092	26.0092	(±1, ±1), (–1, –0.3002), (–0.3002, 1), (0.3002, –1), (1, 0.3002)
9	Exchange	73.6434	23.9848	30.3371	32.8611	(±1, ±1), (–1, 0.5), (–0.5, –1), (0, 0), (0.5, 1), (1, –0.5)*
	Genetic	73.9627	24.0119	29.7853	32.9501	(–1, –0.9926), (–1, 0.4999), (–0.9935, 1), (–0.5003, –1), (–0.0029, –0.0007), (0.5002, 1), (0.9997, –1), (1, –0.4998), (1, 0.9956)
10	Exchange	78.6298	31.2233	33.2633	42.0814	(±1, ±1), (±1, –0.5), (±0.6, 1), (0, –1), (0, 0)*
	Genetic	78.3637	31.7089	32.8844	41.9990	(±1, ±1), (–0.9981, –0.4998), (–0.6022, 0.9932), (0, –0.9983), (0.0132, 0.0275), (0.5997, 0.9974), (1.0000, –0.5005)

* Design generated using the Exchange Algorithm (EA) produced by Srisuradetchai [6]

Also, *D*-efficiencies from GA are greater than or equal to those obtained from EA without missing data.

Furthermore, leave-one-out and Med *D*-efficiencies, interpreted respectively as the mean and median of a design with 1 missing point, are also higher or equal to designs obtained from GA. The *D*-efficiency, Min *D*, Med *D*, or leave-1-out *D* values from GA are at least superior to those from EA.

For *N*-point Min *G*-optimal robust designs, the design points and corresponding properties are shown in Table 2. All designs can be improved by using GA, even a small amount of difference in Min *G* efficiencies.

For each *N*, designs from GA and EA are all different. However, some designs are subtly different. For example, the 8-point design generated by EA is {(±1,±1), (–1,–0.3), (–0.3,1), (0.3,–1), (1,0.3)}and the design generated by GA is {(±1,±1), (–1,–0.3002), (–0.3002,1), (0.3002,–1), (1,0.3002)}. The only difference here is in the *4*th decimal place for coordinates.

Considering *G*-efficiency, Min *G*, Med *G*, and leave-1-out *G* of designs with the same *N*, except the case of $N = 7$, the differences in efficiencies are all less than 0.6. For example, *G*-efficiencies of 9-point designs generated by EA and GA are 73.6434 and 73.9627, respectively. The difference is only 0.3191. Unlike *D* criterion, improving Min *G* does not necessarily increase *G*-efficiency.

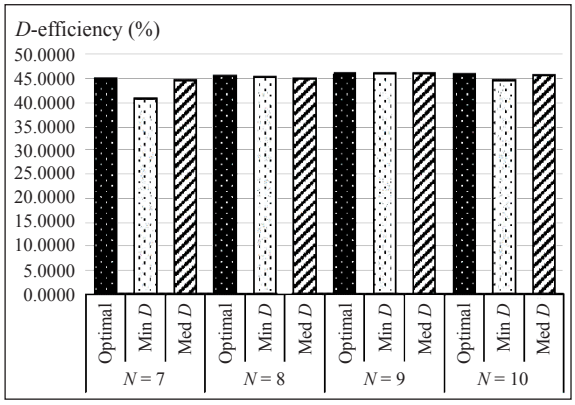

Figure 2: Comparisons of *D*-efficiency values for *N*-point Min *D*-, Med *D*-, and *D*-optimal robust designs.

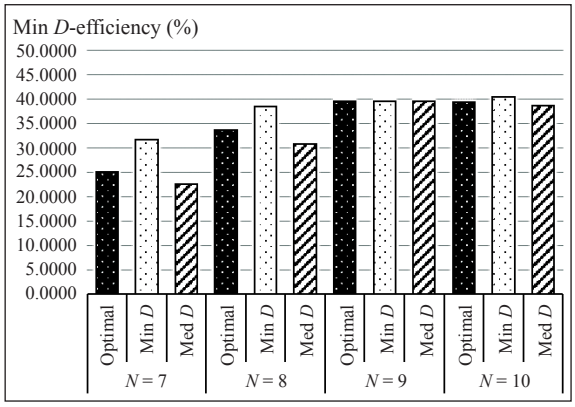

Figure 3: Comparisons of Min *D* values for *N*-point Min *D*-, Med *D*-, and *D*-optimal robust designs.

Table 3: Properties of the *N*-point Min *D*-, Med *D*-, and *D*-optimal robust designs

N	Criteria / Design	*D*-efficiency	Min *D*	Med *D*	Leave-1-out *D*	Design Points
7	Optimal	45.0294	25.1438	39.7711	34.7890	(±1, ±1), (–0.0915, 0.0915), (–0.0675, –1), (1, –0.0675)*
	Min *D*	40.9343	31.6883	31.7902	33.7616	(±1, –1), (–1, 0.1924), (–0.4666, 1), (–0.1775, –0.5799), (1, –0.2747), (1, 1)
	Med *D*	44.7887	22.6502	41.1670	34.1754	(±1, ±1), (–1, 0.165), (–0.1783, 1), (0.1713, –0.1622)
8	Optimal	45.6158	33.6481	41.4995	40.3800	(±1, ±1), (–1, 0.0821), (0, –0.2152), (0, 1), (1, 0.0821)*
	Min *D*	45.4280	38.5145	40.8727	40.8727	(±1, ±1), (±1, 0), (0, ±1)
	Med *D*	45.1028	30.8416	42.2534	39.6587	(±1, ±1), (–1, 0.0385), (–0.0202, –0.2735), (0.0774, 1), (1, 0.4012)
9	Optimal	46.2241	39.5810	45.4280	42.8293	(±1, ±1), (±1, 0), (0, ±1), (0, 0)*
	Min *D*	46.2241	39.5810	45.4280	42.8293	(±1, ±1), (±1, 0), (0, ±1), (0, 0)
	Med *D*	46.2241	39.5810	45.4280	42.8293	(±1, ±1), (±1, 0), (0, ±1), (0, 0)
10	Optimal	45.9888	39.4000	44.6337	43.4774	(±1, ±1), (–1, –0.017), (–0.0993, –1), (–0.017, 1), (0.0243, –0.0243), (1, –1), (1, 0.0993)*
	Min *D*	44.7694	40.4664	42.2662	42.5406	(±1, ±1), (–1, –0.4974), (–1, 1), (–0.1099, 0.0357), (0.2490, –1), (0.4579, 1), (1, –0.1881)
	Med *D*	45.8469	38.7133	44.7346	43.2895	(±1, ±1), (–1, –1), (–1, 0.0347), (–0.0128, 0.1377), (0.0068, 1), (0.0425, –1), (1, –0.0246)

* Design generated by using the Genetic Algorithm (GA) produced by Borkowski [2]

4.2 *Comparing the criteria*

For designs generated by GA, corresponding *D*-efficiency, Min *D*, Med *D*, and leave-1-out *D* values are compared, as shown in Table 3. For $N = 7$, the design points for Min *D*-, Med *D*-, and *D*-optimal designs are different. In this case, the Med *D*- and *D*-optimal designs are quite similar but are different from the Min *D* design in terms of *D*-efficiency, Min *D*, Med *D*, and leave-1-out *D* values. For $N = 8, 9, 10$, the Min *D*-, Med *D*-, and *D*-optimal designs have about the same *D*-efficiencies, as shown in Figure 2. This means that without a missing point, Min *D*- and Med *D*-optimal robust designs are nearly *D*-optimal. Also, *D*-efficiency of Med *D*-optimal designs tends to be larger than that of Min *D*-optimal designs. From Figure 3, Med *D*-optimal robust exact designs tend to give the smallest Min *D* efficiency value, i.e.

$\text{Min } D_{\text{Min } D \text{ Design}} \geq \text{Min } D_{D \text{ Design}} \geq \text{Min } D_{\text{Med } D \text{ Design}}$.

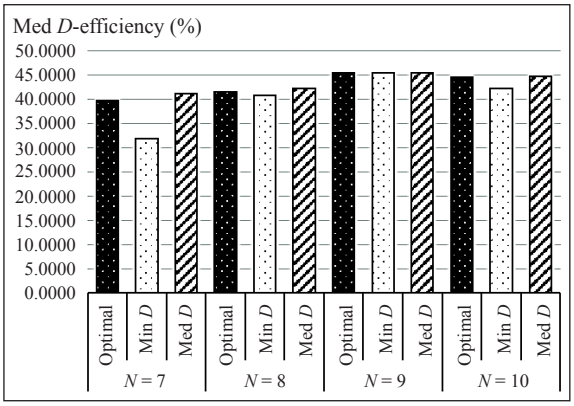

Figure 4: Comparisons of Med *D* values for *N*-point Min *D*-, Med *D*-, and *D*-optimal robust designs.

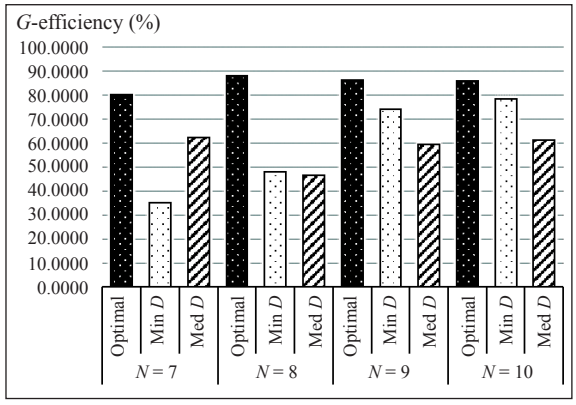

Figure 5: Comparisons of *G*-efficiency values for *N*-point Min *G*-, Med *G*-, and *G*-optimal robust designs.

Table 4: Properties of the *N*-point Min *G*-, Med *G*-, and *G*-optimal robust designs

N	Criteria / Design	*G*-efficiency	Min *G*	Med *G*	Leave-1-out *G*	Design Points
7	Optimal	80.1029	0.0000	21.5623	15.8832	(–1, –0.6031), (–1, 0.9378), (–0.6031, –1), (–0.0507, 0.773), (0.773, –0.0507), (0.9378, –1), (1, 1)*
	Min *G*	35.0715	9.9635	10.1688	14.2102	(–1, –0.9949), (–1, 0.995), (–0.5584, 0.3226), (–0.2848, –1), (0.1984, 1), (0.9692, –1), (1, 0.3863)
	Med *G*	62.4056	0.5206	31.4632	18.5443	(–1, 1), (–1, –0.9985), (–0.998, 0.1678), (–0.2152, 1), (0.121, –0.0913), (0.9812, –0.9814), (1, 1)
8	Optimal	87.9430	14.7921	28.7295	29.2077	(±1, ±1), (–1, 0.0522), (–0.0633, –0.8246), (0.0633, 0.8246), (1, –0.0522)*
	Min *G*	47.8695	18.3569	26.0092	26.0092	(±1, ±1), (–1, –0.3002), (–0.3002, 1), (0.3002, –1), (1, 0.3002)
	Med *G*	46.6297	0.1325	43.6185	28.4287	(–1, –0.6427), (–0.9505, 1), (–0.731, –1), (–0.7174, –0.0059), (–0.0445, –1), (–0.0445, 0.4281), (0.9053, 0.8313), (1, –0.9572)
9	Optimal	86.3165	22.1432	37.7760	35.2833	(±1, ±1), (–1, 0.4202), (–0.4202, –1), (0, 0), (0.4202, 1), (1, –0.4202)*
	Min *G*	73.9627	24.0119	29.7853	32.9501	(–1, –0.9926), (–1, 0.4999), (–0.9935, 1), (–0.5003, –1), (–0.0029, –0.0007), (0.5002, 1), (0.9997, –1), (1, –0.4998), (1, 0.9956)
	Med *G*	59.2783	0.0067	66.0352	43.9788	(–1, 0.5225), (–1, 0.5477), (–0.8947, –1), (–0.4805, 0.9931), (–0.3513, 0.9931), (0.1415, 0), (0.2213, –0.0864), (1, –1), (1, 0.9012)
10	Optimal	85.9260	29.4805	36.6922	42.3716	(±1, ±1), (±1, –0.43), (±0.564, 1), (0, –1), (0, 0.1766)*
	Min *G*	78.3637	31.7089	32.8844	41.9990	(±1, ±1), (–0.9981, –0.4998), (–0.6022, 0.9932), (0, –0.9983), (0.0132, 0.0275), (0.5997, 0.9974), (1.0000, –0.5005)
	Med *G*	61.1967	1.3296	67.7520	46.9870	(±1, –1), (–1, 0.5073), (–1, 0.5754), (–0.518, 1), (–0.4859, 0.9941), (–0.0178, –0.9288), (0.5528, 0.0111), (0.5639, 0), (1, 0.9903)

*Design generated using the Genetic Algorithm (GA) produced by Borkowski [2]

Figure 4 indicates that the Med *D* values for *D*-optimal and Med *D*-optimal robust designs are about the same. Also, the Med *D* values for the Min *D*-optimal designs are smaller than those for the *D*-optimal designs, i.e., Med $D_{\text{Med } D \text{ Design}} \geq$ Med $D_{D \text{ Design}} \geq D_{\text{Min } D \text{ Design}}$. As the sample size increases, the *D*-efficiency, Min *D*, Med *D*, or leave-1-out *D* values for all *N*-point designs remain about the same, e.g., in the case of $N = 10$.

Table 4 summarizes the properties of 7-, 8-, 9-, and 10-point designs constructed using Min *G*-, Med *G*-, and *G*-optimality criteria. It was observed that the *G*-efficiency values of the 7-point Min *G*-optimal design were much smaller than those of Med *G*-optimal design, as shown in Figure 5. However, it is about the same value in the 8-point design. For the 9- and 10-point Min *G*-optimal designs, the *G*-efficiencies are higher than those of Med *G*-optimal designs. For Min *G*

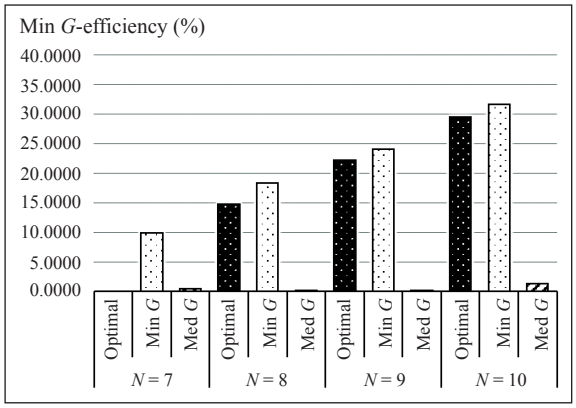

Figure 6: Comparisons of Min *G* values for *N*-point Min *G*-, Med *G*-, and *G*-optimal robust designs.

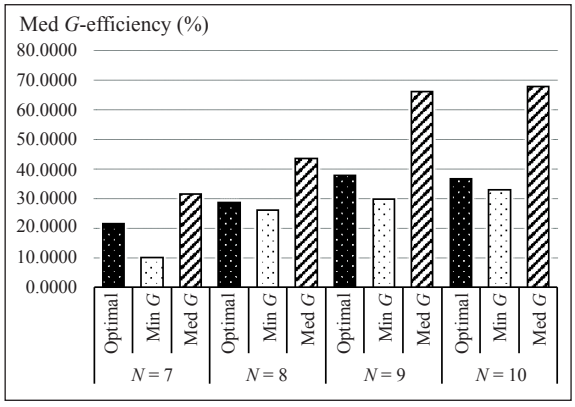

Figure 7: Comparisons of Med *G* values for *N*-point Min *G*-, Med *G*-, and *G*-optimal robust designs.

comparisons in Figure 6, it is obvious that Min *G* of Med *G*-optimal designs is much smaller than that of Min *G*- and *G*-optimal designs, except in the case of $N = 7$. For example, the 9-point *G*-optimal design has Min *G* of 22.143 and Med *G* of 37.776, while the Med *G*-optimal design has Min *G* of 0.0067 and Med *G* of 66.0352. Also, Figure 7 shows that the difference in Med *G* values for optimal *G*- and Med *G*-optimal designs tends to become larger as the sample size increases. The Min *G*-optimal design always produces the smallest Med *G*-efficiency.

5 Conclusion and Discussion

When comparing EA and GA designs, Min *D*- or Min *G*-efficiencies from GA are always higher than or equal to those from EA. For 8- and 9-point Min *D* designs, GA and EA produce the same design points. Thus, it can be said that the resulting designs are Min *D*-optimal in a continuous design space.

The proposed Med *D*-optimal robust and *D*-optimal designs have similar *D*-efficiencies. If the worst case happens, *D*-optimal designs are suggested to be used as their Min *D* values are close to those of Min *D*-optimal robust designs. However in case of a general missing point, the Med *D*-optimal robust design would be superior to the Min *D*-optimal design, especially for a very small design such as $N = 7$.

For *G*-optimality, the Med *G*-optimal robust designs would not be recommended if the worst case happened. In such a case, *G*-optimal designs would give Min *G* much larger than that of Med *G* designs. However, $|\mathbf{X}^T\mathbf{X}|$ of Med *G*-optimal robust design is not singular, unlike the *G*-optimal design with $N = 7$. Without a missing point, Min *G*-optimal designs would be recommended, while Med *G*-optimal designs are far better than *G*-optimal designs for a general missing point (not the worst case) although a sample size increases.

Acknowledgements

The authors gratefully acknowledge the financial support provided by Thammasat University Research Fund under the TU Research Scholar, Contract No. 1/3/2559.

References

[1] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 4th ed., Hoboken, NJ: Wiley, 2016.

[2] J. J. Borkowski, "Using a genetic algorithm to generate small exact response surface designs," *Journal of Probability and Statistics Science*, vol. 1, no. 1, pp. 65–88, Feb. 2003.

[3] A. Wald, "On the efficient design of statistical investigations," *The Annals of Mathematical Statistics*, vol. 14, no. 2, pp. 134–140, Jun. 1943.

[4] E. D. Castillo, *Process optimization: A statistical approach*. New York: Springer US, 2007.

[5] K. Smith, "On the standard deviations of adjusted and interpolated values of an observed

polynomial function and its constants and guidance they give towards a proper choice of distribution of observations," *Biometrika*, vol. 12, no. 1/2, pp. 1–85, Nov. 1918.

[6] P. Srisuradetchai, "Robust response surface designs against missing observations," Ph.D. dissertation, Montana State University, Bozeman, MT, 2015.

[7] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MA: A Bradford Book, 1992.

[8] *Handbook of Genetic Algorithms*, 1st ed., New York: Van Nostrand Reinhold, 1991.