A General Method for Constructing Efficient Choice Designs

Klaus Zwerina

Joel Huber

Warren F. Kuhfeld *

September 1996

* Klaus Zwerina is a consultant at BASF AG, Ludwigshafen, Germany. Joel Huber is Professor of Marketing, Fuqua School of Business, Duke University. Warren F. Kuhfeld is Manager, Multivariate Models R&D, Statistical Research and Development, SAS Institute Inc. We would like to thank Jim Bettman and Richard Johnson for their helpful comments on an earlier version of this paper.

A General Method for Constructing Efficient Choice Designs

ABSTRACT

Researchers have traditionally built choice designs using extensions of concepts from the general linear design literature. We show that a computerized search strategy can generate efficient choice designs with standard personal computers. This approach holds three important advantages over previous design strategies. First, it allows the incorporation of anticipated model parameters, thereby increasing design efficiency and considerably reducing the number of required choices. Second, complex choice designs can be easily generated, allowing researchers to conduct choice experiments that more closely mirror actual market conditions. Finally, researchers can explore model and design modifications and examine trade-offs between a design's statistical benefits and its operational and behavioral costs.

INTRODUCTION

Discrete choice experiments are becoming increasingly popular in marketing, economics, and transportation. These experiments enable researchers to model choice in an explicit competitive context, thus realistically emulating market decisions. A choice design consists of choice sets composed of several alternatives, each defined as combinations of different attribute levels. A good choice design is efficient, meaning that the parameters of the choice model are estimated with maximum precision.

A number of methods have been suggested for building choice designs (Anderson and Wiley 1992, Bunch, Louviere, and Anderson 1996, Krieger and Green 1991, Kuhfeld 2000, Lazari and Anderson 1994, Louviere and Woodworth 1983). Most of the methods use extensions of standard linear experimental designs (Addelman 1962, Green 1974). However, the use of linear designs in choice experiments may be nonoptimal due to two well-known differences between linear and choice models. First, probabilistic choice models are nonlinear in the parameters, implying that the statistical efficiency of a choice design depends on an (unknown) parameter vector. This property implies the need to bring anticipated parameter values in choice designs. Second, choice design efficiency depends both on the creation of appropriate profiles and properly placing them into several choice sets. For example, in a linear design, the order of the 16 profiles in a conjoint exercise does not affect its formal efficiency, whereas the efficiency of the same 16 profiles broken into four choice sets depends critically on the grouping. Despite its limitations, linear design theory has been used to produce satisfactory choice designs for many years, drawing on readily available tables and processes. Such carefully selected linear designs are reasonable, general-purpose choice designs, but are generally not optimal in a statistical sense.

We present a general strategy for the computerized construction of efficient choice designs. This contribution can be viewed as an extension of the work of Kuhfeld, Tobias, and Garratt (1994) and of Huber and Zwerina (1996). Kuhfeld et al. recommended using a search algorithm to find efficient *linear* designs. Huber and Zwerina show how to modify *choice*

designs using anticipated model parameters in order to improve design efficiency. We adapt the optimization procedure outlined in Kuhfeld et al. to the principles of choice design efficiency described by Huber and Zwerina. Our approach holds several important advantages over previous choice design strategies. It (1) optimizes the "correct" criterion of minimizing estimation error rather than following linear design principles, (2) it can generate choice designs that accommodate any anticipated parameter vector, (3) it can accommodate virtually any level of model complexity, and finally (4) it can be built using widely available software. To illustrate, we include a SAS/IML program that generates relatively simple choice designs. This program can be easily generalized to handle far more complex problems.

The paper begins with a review of criteria for efficient choice designs and illustrates how they can be built with a computer. Then, beginning with simple designs, we illustrate how the algorithm works and how our linear design intuition must be changed when coping with choice designs. Next, we generate more complex choice designs and show how to evaluate the impact on efficiency of different design and model modifications. We conclude with a discussion of the proposed choice design approach and directions for future research.

CRITERIA FOR CHOICE DESIGN EFFICIENCY

Measure of Choice Design Efficiency

First, we derive a measure of efficiency in choice designs from the well-known multinomial logit model (McFadden 1974). This model assumes that consumers make choices among alternatives that maximize their perceived utility, u, given by

$$u = \mathbf{x}_i \boldsymbol{\beta} + e, \tag{1}$$

where $\mathbf{x}_i$ is a row vector of attributes characterizing alternative i, $\boldsymbol{\beta}$ is a column vector of K weights associated with these attributes, and e is an error term that captures unobserved variations in utility. Suppose that there are N choice sets, C_n, indexed by $n=1, 2, \ldots, N$, where each choice set is characterized by a set of alternatives $C_n = \{x_{1n}, \mathrm{K}\ , x_{J_n n}\}$. If the errors, e, are independently and identically Gumbel distributed, then it can be shown that the probability of choosing an alternative i from a choice set C_n is

$$P_{in}(\mathbf{X}_n, \beta) = \frac{e^{x_{in}\beta}}{\sum_{j=1}^{J_n} e^{x_{jn}\beta}}, \tag{2}$$

where $\mathbf{X}_n$ is a matrix that consists of J_n row vectors, each describing the characteristics of the alternatives, $\mathbf{x}_{jn}$. The vertical concatenation of the $\mathbf{X}_n$ matrices is called a choice design matrix $\mathbf{X}$.

The task of the analyst is to find a parameter estimate for $\boldsymbol{\beta}$ in Equation (2) that maximizes the likelihood given the data. Under very general conditions, the maximum likelihood estimator is consistent and asymptotically normal with covariance matrix

$$\Sigma = (\mathbf{Z}'\mathbf{P}\mathbf{Z})^{-1} = [\sum_{n=1}^{N}\sum_{j=1}^{J_n} \mathbf{z}'_{jn} P_{jn} \mathbf{z}_{jn}]^{-1}, \tag{3}$$

$$\text{where } \mathbf{z}_{jn} = \mathbf{x}_{jn} - \sum_{i=1}^{J_n} \mathbf{x}_{in} P_{in}.$$

Equation (3) reveals some important properties of (nonlinear) choice models. In linear models centering occurs across all profiles whereas in choice models, centering occurs within choice sets. This shows that in choice designs both the profile selection and the assignment of profiles to choice sets affects the covariance matrix. Moreover, in linear models, the covariance matrix does not depend on the true parameter vector, whereas in choice models the probabilities, P_{jn}, are functions of $\boldsymbol{\beta}$ and hence the covariance matrix. Assuming $\beta = 0$ simplifies the design problem, however, Huber and Zwerina (1996) recently demonstrated that this assumption may be costly. They showed that incorrectly assuming that $\beta = 0$ may require from 10% to 50% more respondents than those built from reasonably anticipated parameters.

The goal in choice designs is to define a group of choice sets, given anticipated $\boldsymbol{\beta}$, that minimizes the "size" of the covariance matrix, Σ, defined in Equation (3). There are various summary measures of error size that can be derived from the covariance matrix (see, e.g., Raktoe, Hedayat, and Federer 1981). Perhaps the most intuitive summary measure is the average variance around the estimated parameters of a model. This measure is referred to in the literature as A-efficiency or its inversely related counterpart,

$$A - error = trace(\Sigma) / K, \tag{4}$$

where K is the number of parameters. Two problems with this measure limit its suitability as an overall measure of design efficiency. First, relative A-error is not invariant under (nonsingular) recodings of the design matrix, i.e., design efficiency depends on the type of coding. Second, it is computationally expensive to update. A related measure,

$$D - error = |\Sigma|^{1/K}, \tag{5}$$

is based on the determinant as opposed to the trace of the covariance matrix. D-error is computationally efficient to update, and the ratios of D-errors are invariant under different codings of the design matrix. Since A-error is the arithmetic mean and D-error is the geometric mean of the eigenvalues of Σ, they are generally highly correlated. D-error thereby provides a reasonable way to find designs that are "good" on alternative criteria. For example, if A-error is the ultimate criterion, we can first minimize D-error and then select the design with minimum A-error rather than minimizing A-error directly. For these reasons, D-error (or its inverse, D-efficiency or D-optimality) is the most common criterion for evaluating linear designs and we advocate it as a criterion for choice designs.

Next, we discuss four principles of choice design efficiency defined by Huber and Zwerina (1996). Choice designs that satisfy these principles are optimal, however, these principles are only satisfied for a few special cases and under quite restrictive assumptions. The

principles of orthogonality and balance that figured so prominently in linear designs remain important in understanding what makes a good choice design, but they will be less useful in generating one.

Four Principles of Efficient Choice Designs

Huber and Zwerina (1996) identify four principles which when jointly satisfied indicate that a design has minimal D-error. These principles are orthogonality, level balance, minimal overlap, and utility balance. *Orthogonality* is satisfied when the levels of each attribute vary independently of one another. *Level balance* is satisfied when the levels of each attribute appear with equal frequency. *Minimal overlap* is satisfied when the alternatives within each choice set have nonoverlapping attribute levels. *Utility balance* is satisfied when the utilities of alternatives within choice sets are the same, i.e., the design will be more efficient as the expected probabilities within a choice set C_n among J_n alternatives approach $1/J_n$.

These principles are useful in understanding what makes a choice design efficient, and improving any principle, holding the others constant, improves efficiency. However, for most combinations of attributes, levels, alternatives, and assumed parameter vectors, it is impossible to create a design that satisfies these principles. The proposed approach does not build choice designs from these formal principles, but instead uses a computer to directly minimize D-error. As a result, these principles may only be approximately satisfied in our designs, but they will generally be more efficient than those built directly from the principles.

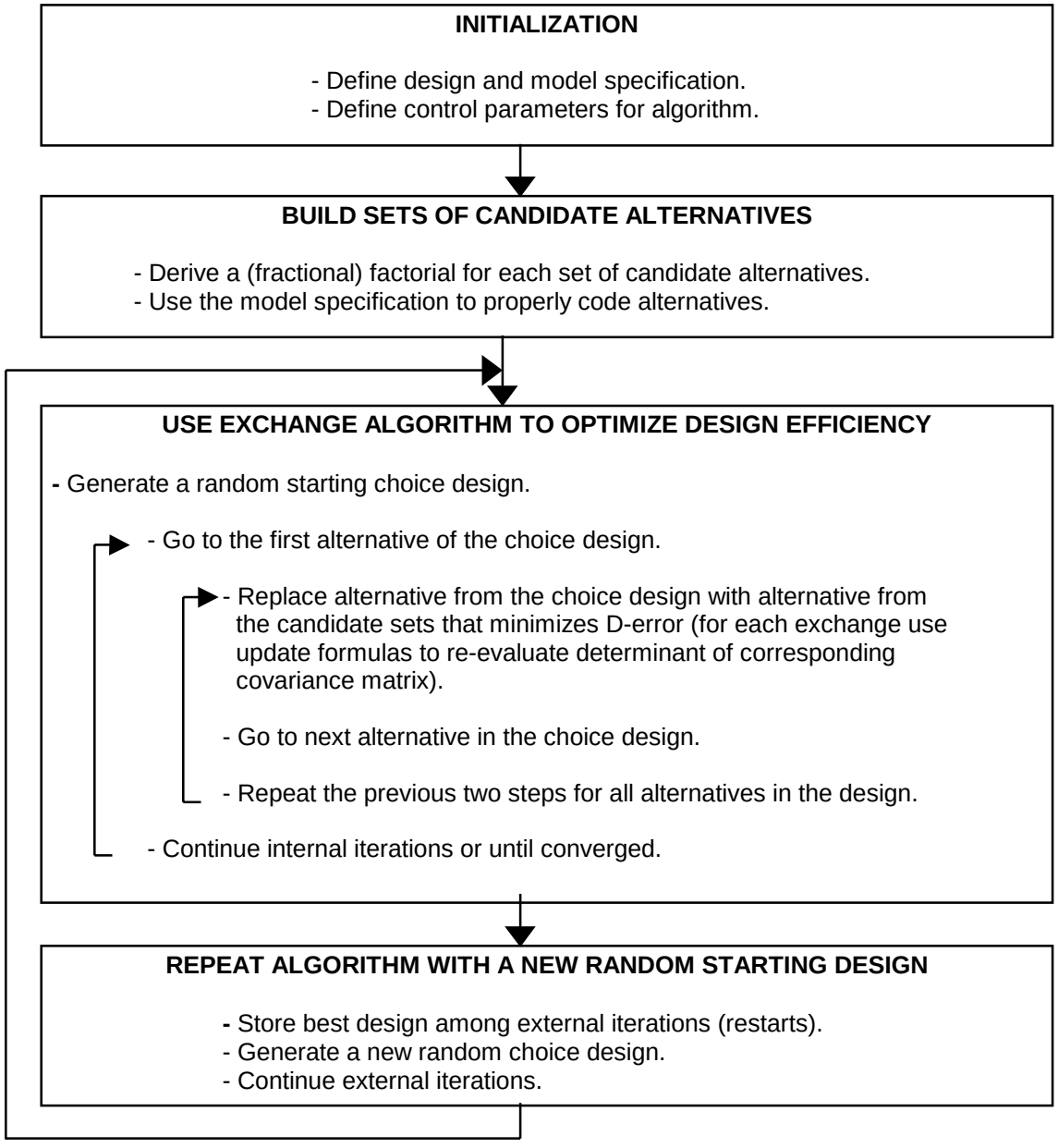
A GENERAL METHOD FOR EFFICIENT CHOICE DESIGNS

Figure 1 provides a flowchart of the proposed design approach. The critical aspect of this approach involves an adaptation of Cook and Nachtsheim's (1980) modification of the Federov (1972) algorithm that has successfully been used to generate efficient *linear* designs (e.g., Cook and Nachtsheim 1980, Kuhfeld et al. 1994). We will first describe the proposed choice design approach conceptually and then define the details in a context of a particular search.

The process begins by building a *candidate set*, which is a list of potential alternatives. A random selection of these alternatives is the *starting design*. The algorithm alters the starting design by exchanging its alternatives with the candidate alternatives. The algorithm finds the best exchange (if one exists) for the first alternative in the starting design. The first iteration is completed once the algorithm has sequentially found the best exchanges for all of the alternatives in the starting design. After that, the process moves back to the first alternative and continues until no substantial efficiency improvement is possible. To avoid poor local optima, the whole process can be restarted with different random starting designs and the most efficient design is selected. For example, if there are 300 alternatives in the candidate set and 50 alternatives in the choice design, then each iteration requires testing 15,000 possible exchanges, which is a reasonable problem on today's desktop computers and workstations. While there is no guarantee that it will converge to an optimal design, our experience with relatively small problems suggests that the algorithm works very well.

Figure 1

FLOWCHART OF ALGORITHM FOR CONSTRUCTING EFFICIENT CHOICE DESIGNS



To illustrate the process we first generate choice designs for simple models that reveal the characteristics of efficient choice designs. In examining these simple designs, our focus is on the benefits and the insights that derive from using this approach. Then, we apply the approach to more complex design problems, such as alternative-specific designs and designs with constant alternatives. As we illustrate more complex designs, we will focus on the use of the approach, *per se*. We provide illustrative computer code in the appendix.

CHOICE DESIGN APPLICATIONS

Simple Designs

Generic Models. The simplest choice models involve alternatives described by generic attributes. The utility functions for these models consist of attribute parameters that are the same for all alternatives, for example, a common price slope across all alternatives. Generic designs are appealing because they are simple and analogous to main-effects conjoint experiments. Bunch et al. (1996) evaluate ways to generate generic choice designs and show that shifted or cyclic designs generally have superior efficiency compared with other strategies for generating main effects designs. These shifted designs use an orthogonal fractional factorial to provide the "seed" alternatives for each choice set. Subsequent alternatives within a choice set are cyclically generated. The attribute levels of the new alternatives add one to the level of the previous alternative until it is at its highest level, at which point the assignment re-cycles to the lowest level.

For certain families of plans and assuming that all coefficients are zero, these shifted designs satisfy all four principles, and thus are optimal.[1] For example, consider a choice experiment with three attributes, each at three levels, defining three alternatives in each of nine choice sets. The left-hand panel of Table 1 shows a plan using the Bunch et al. (1996) method.

In this special case, all four efficiency principles are perfectly satisfied. Level balance is satisfied since each level occurs in precisely 1/3 of the cases, and orthogonality can be confirmed by noting the all pairs of attribute levels occur in precisely 1/9 of the attributes (Addelman 1962). There is perfect minimal overlap since each level occurs exactly once in each choice set, and finally, utility balance is trivially satisfied with the assumption that β=0. More formally, it is useful to examine the covariance matrix of the (effects-coded) parameters, reported in the first panel of Table 2. The equal variances across attributes and the zero covariances across attributes both indicate optimality.

A simple design such as this could have been built from our algorithm, although using a standard orthogonal array and cyclic permutations ensured optimality. Our next example, encompassing a model with just one interaction term, illustrates the case when a computerized search is very useful in finding a statistically efficient design.

Estimating an A×B Interaction. For the previous example with nine choice sets, let us assume that the researcher is confident that there are no A×C or B×C interactions, but the A×B interaction must be estimated. The middle panel of Table 1 shows the best design we were able to find which includes this one interaction. Note that in this design, the principle of minimal overlap on attributes A and B is violated, in that attribute levels are frequently repeated within a set. In general, interactions *require* overlap of attribute levels to produce the contrasts necessary to estimate the effects.

[1] We were not able to analytically prove this, but after examining scores of designs, we have never found more efficient designs than those that satisfy all four principles.

Table 1
MAIN EFFECTS AND A×B-INTERACTION EFFECTS CHOICE DESIGN

		β_0-Efficient Main Effects Design (β_0={0 0 0 0 0 0})			β_0-Efficient Interaction Effects Design (β_0={0 0 0 0 0 0 0 0 0})				β_1-Efficient Interaction Effects Design (β_1={-1 0 -1 0 -1 0 0 0 0})			
		Attr. A	Attr. B	Attr. C	Attr. A	Attr. B	Attr. C	$p(\beta_1)$	Attr. A	Attr. B	Attr. C	$p(\beta_1)$
Set 1	Alt. I	1	1	1	2	3	2	.495	2	1	3	.422
	Alt. II	2	2	2	2	2	3	.495	1	3	2	.422
	Alt. III	3	3	3	1	1	1	.009	3	1	1	.155
Set 2	Alt. I	1	2	2	3	1	1	.155	3	2	2	.422
	Alt. II	2	3	3	2	2	2	.422	2	1	3	.155
	Alt. III	3	1	1	1	2	3	.422	3	3	1	.422
Set 3	Alt. I	1	3	3	1	1	2	.042	2	2	3	.155
	Alt. II	2	1	1	1	3	1	.114	3	3	2	.422
	Alt. III	3	2	2	3	1	3	.844	2	3	3	.422
Set 4	Alt. I	2	1	3	2	1	2	.018	2	1	2	.422
	Alt. II	3	2	1	3	3	3	.965	1	1	3	.422
	Alt. III	1	3	2	2	2	1	.018	1	2	1	.155
Set 5	Alt. I	2	2	1	1	3	3	.245	3	1	2	.422
	Alt. II	3	3	2	3	3	2	.665	1	1	3	.155
	Alt. III	1	1	3	2	3	1	.090	3	2	1	.422
Set 6	Alt. I	2	3	2	2	1	3	.468	1	3	3	.422
	Alt. II	3	1	3	1	2	1	.063	2	3	2	.422
	Alt. III	1	2	1	1	3	2	.468	3	2	1	.155
Set 7	Alt. I	3	1	2	3	2	3	.665	1	2	1	.212
	Alt. II	1	2	3	3	3	1	.245	2	2	1	.576
	Alt. III	2	3	1	3	1	2	.090	1	1	2	.212
Set 8	Alt. I	3	2	3	1	2	2	.042	1	2	3	.576
	Alt. II	1	3	1	2	3	3	.844	1	3	1	.212
	Alt. III	2	1	2	3	2	1	.114	3	1	1	.212
Set 9	Alt. I	3	3	1	1	1	3	.114	2	2	2	.212
	Alt. II	1	1	2	2	1	1	.042	3	1	3	.576
	Alt. III	2	2	3	3	2	2	.844	2	3	1	.212
							avemaxp=	.690			avemaxp=	.474
		D-error(β_0) = .192			D-error(β_0) = .306				D-error(β_0) = .365			
					D-error(β_1) = .630				D-error(β_1) = .399			

The covariance matrix of this design, depicted in the lower half of Table 2, highlights the effects of incorporating the A×B interaction. Violating minimal overlap permits the estimation of the A×B interaction by sacrificing efficiency on the main effects of attribute A and B, reflected in higher variances of the main effects estimates (a1, a2, b1, and b2). The D-error of the main effect estimates increases by 24%, from .192 to .239, and the covariances across attributes A and B are no longer zero. Note also that the errors around attribute C are unchanged—they are unaffected by the A×B interaction, indicating that the algorithm was able to find a design that allowed the A×B interaction to be uncorrelated with C.

There are several important lessons from this simple example. First, it illustrates that a design that is "perfect" for one model may be far from optimal for a slightly different model. Adding one interaction strongly altered the covariance matrix, so efficient designs generally violate the formal principles. Second, the example shows that estimating new interactions is not without cost; being able to estimate one interaction increased by 24% the error on the main effects. Finally, the trade-off of efficiency with estimability demonstrates one of the primary benefits of this approach—it allows the analyst to understand the efficiency implications of changes in the design structure and/or model specification. This use of the approach will be illustrated again in the context of more complex choice designs.

Table 2
COVARIANCE MATRIX OF MAIN EFFECTS AND A×B-INTERACTION EFFECTS CHOICE DESIGN

β_0-Efficient Main Effects Design

	a1	*a2*	*b1*	*b2*	*c1*	*c2*
a1	.222	-.111	.000	.000	.000	.000
a2	-.111	.222	.000	.000	.000	.000
b1	.000	.000	.222	-.111	.000	.000
b2	.000	.000	-.111	.222	.000	.000
c1	.000	.000	.000	.000	**.222**	**-.111**
c1	.000	.000	.000	.000	**-.111**	**.222**

D-error(β_0)=.192

β_0-Efficient Interaction Effects Design

	a1	*a2*	*b1*	*b2*	*c1*	*c2*	*ab11*	*ab12*	*ab21*	*ab22*
a1	.296	-.130	.019	-.019	.000	.000	-.037	.000	.000	-.019
a2	-.130	.296	-.019	.019	.000	.000	.037	.000	.000	.019
b1	.019	-.019	.296	-.130	.000	.000	.019	-.056	.000	.037
b2	-.019	.019	-.130	.296	.000	.000	-.019	.056	.000	-.037
c1	.000	.000	.000	.000	**.222**	**-.111**	.000	.000	.000	.000
c2	.000	.000	.000	.000	**-.111**	**.222**	.000	.000	.000	.000
ab11	-.037	.037	.019	-.019	.000	.000	.630	-.333	-.333	.148
ab12	.000	.000	-.056	.056	.000	.000	-.333	.556	.167	-.278
ab21	.000	.000	.000	.000	.000	.000	-.333	.167	.667	-.333
ab22	-.019	.019	.037	-.037	.000	.000	.148	-.278	-.333	.630

D-error(β_0) of main effects = .239
D-error(β_0) of all effects = .306

The Impact of Non-Zero Betas. The preceding discussion has assumed that the true parameters are zero. This assumption is justified when there is very little information about the model parameters; however, typically the analyst has some information on the relative importance of attributes or the relative value of their levels (Huber and Zwerina 1996). To show the potential gain that can come from nonzero parameters, assume that the anticipated partworths of the main effects for the three level attributes discussed above are not 0,0,0, but -1, 0, 1, while the A×B-interaction effect continues to have zero parameters.[2] Calling the new parameter vector $\boldsymbol{\beta}_1$ to distinguish it from the zero parameter vector, $\boldsymbol{\beta}_0$, the third panel of Table 1 displays the efficient design using these parameters. This new design has a D-error($\boldsymbol{\beta}_1$) of .399. However, if instead we had used the design in the center panel, its error given $\boldsymbol{\beta}_1$ is true would have been .630, implying that 37% (1-.399/.630) fewer respondents are needed for the "utility balanced" over the "utility neutral" design.

Comparing the last two panels in Table 1 reveals how the algorithm used the anticipated nonzero parameters to produce a more efficient design. As an index of utility balance, we calculated the average of the maximum within-choice-set choice probabilities (avemaxp). The smaller this index the harder is the average choice task and the greater is "utility balance". We can see, by using $\boldsymbol{\beta}_1$, the new design is more utility balanced than the previous design, which results in an average maximum probability of .474 compared with one of .690. We also see that the increase in utility balance sacrifices somewhat the three formal principles, reflected in an

[2] We assume for simplicity that the interaction has parameter values of zero. Note, this also produces minimal variance of estimates around zero, implying greatest power of a test in the region of that null hypothesis.

increase of D-error(β_0) from .306 to .365. The new design does not have perfect orthogonality, level balance, utility balance, or minimal overlap, but it is more efficient than any design that is perfect on any of those criteria.

More Complex Choice Designs

The proposed algorithm is very general and can be applied to virtually any level of design complexity. We will use it next to generate an *alternative-specific* choice design, which has a separate set of parameters for each alternative. Suppose, the researcher is interested in simulating the market behavior of three brands, Coke, Pepsi, and RC Cola, with the attribute combinations shown in Table 3.

This kind of choice experiment, which we call a *market emulation study,* is quite different from the generic choice design presented previously. In a market emulation study, emphasis is on predicting the impact of brand, flavor, and container decisions in the context of a realistic market place offering. What this kind of study gains in realism, it loses in the interpretability of its results. For example, since each brand only occurs at specific prices, it is much harder to disentangle the independent effects of brand and price. These designs are, however, useful in assessing the managerially critical question of the impact of, say, a 60 cent drop in the price of Coke's 16 oz case in a realistic competitive configuration.

Table 3
ATTRIBUTES/LEVELS FOR AN ALTERNATIVE-SPECIFIC CHOICE EXPERIMENT

Attributes	*Alternative-Specific Levels*		
	Coke	*Pepsi*	*RC Cola*
Price per case	\$5.69	\$5.39	\$4.49
	\$6.89	\$5.99	\$5.39
	\$7.49	\$6.59	\$5.99
Container	12 oz cans	12 oz cans	12 oz cans
	10 oz bottle	10 oz bottle	16 oz bottle
	16 oz bottle	18 oz bottle	22 oz bottle
Flavor	Regular	Regular	Regular
	Cherry Coke	Pepsi Lite	Cherry
	Diet Coke	Diet Pepsi	Diet

Since we assume that the impact of price depends on the brand to which it is attached, it is important that the impact of price be estimable within each brand.[3] Further, let us assume that the reaction to price additionally depends on the number of ounces, so that it is necessary to estimate the brand×price×container interaction. Using standard ANOVA-coding, these assumptions require four main effects (brand, flavor, container, and price for 8 df), four two-way

[3] The assumption that price has a different impact depending on the brand is testable. The ability to make that test is just one of the advantages of these choice designs.

interactions (brand×price, brand×flavor, brand×container, and price×container for 16 df), and one three-way interaction (brand×price×container for 8 df), resulting in a total of 32 parameters.[4]

Suppose we want to precisely estimate these effects with a choice design consisting of 27 choice sets each composed of three alternatives.[5] The candidate set of alternatives comprises the 3^4=81 possible alternatives, and the initial design is a random selection from these. The algorithm exchanges alternatives between the candidate set and the starting design until the efficiency gain becomes negligible. In the example with 27 choice sets and 32 parameters, D-error is .167. This statistic provides a baseline for evaluating other related designs, which we will generate in the following section.

Evaluating Design Modifications. The proposed approach can be used to evaluate design modifications. Typically, efficiency is meaningful within a relatively narrow family of designs, limited to a particular attribute structure, model specification, and number of alternatives per choice set. For many applications, optimizing a design within such a narrow design family is too restrictive. Most analysts are not tightly bound to a particular number of alternatives per choice set or even particular attributes, but are interested in exploring the impact of changes in these specifications on the precision of the parameter estimates. We will demonstrate how comparing designs across design families allows a reasoned trade-off of design structure against estimation precision.

Consider the following questions an analyst might ask concerning the alternative-specific choice design just presented.

1. How much does efficiency increase if 54 choice sets are used instead of two replications of 27 choice sets?
2. What is the efficiency loss if each of the brands (Coke, Pepsi, RC) must be present in a choice set?
3. What is the gain in efficiency if a fourth alternative is added to each choice set?
4. What happens to efficiency if this fourth alternative is constant (e.g., “keep on shopping”)?

The first question assesses the benefit of building a design with 54 choice sets rather than using the original 27 choice sets twice. As Table 4 shows, specifying twice as many choice sets produces a D-error of .079 compared with .084 (=.167/2) for two independent runs of the 27 choice set design. This relatively small 6% benefit in efficiency indicates that the original 27 choice set design, while highly fractionated, appears to have suffered little due to this fact.

[4] We need the fourth two-way interaction, price×container, to be able to estimate the three-way interaction brand×price×container. Of course, there are many other ways of coding a design.

[5] The appendix contains a SAS/IML program that performs the search for this design. Focusing on the principles of the algorithm, the program was deliberately kept simple, specific, and small. A general macro for searching choice designs along with the code used in this paper is available from the authors.

Table 4
IMPACT OF DESIGN MODIFICATIONS ON D-ERROR

Design Modification	*D-error*	*Efficiency per Choice Set*	*Comments*
27 sets, 3 alternatives per set.	.167	100 %	Original design.
Double the number of sets.	.079	106 %	Limited benefit from doubling the number of sets.
Require each alternative to contain one of each brand.	.175	95 %	Shows minor cost of constraining a design.
Add a fourth alternative.	.144	116 %	Diminishing returns from adding additional alternatives.
Fourth alternative is constant.	.195	86 %	Design is less efficient because constant alternative is chosen 25% of the time.

The second question evaluates the impact of constraints on the choice sets that respondents face. The original design often paired the same brand against itself within a choice set. For example, a choice set with Coke in a 12 oz bottle for $5.69 per case might include Coke in a 16 oz bottle for $7.49 per case. For managerial reasons it might be desirable to have each brand (Coke, Pepsi, RC) represented in every set of three alternatives. To examine the cost of this constraint, Coke is assigned to the first alternative, Pepsi to the second alternative, and RC to the third alternative within each of the 27 choice sets. With this constraint, the D-error is .175. This relatively moderate decrease in efficiency of 5% should be acceptable if there are managerially-based reasons to constrain the choice sets.

The third question investigates the benefits of adding a fourth alternative to each choice set. This change increases by 25% the number of alternatives, although the marginal effect of an additional alternative should not be as great. With this modification, D-error becomes .144, producing a 16% efficiency gain over three alternatives per choice set. The decision whether to include a fourth alternative now pits the analyst's appraisal of the trade-off between the value of this 16% efficiency gain and the cost in respondent time and reliability.

What happens if this fourth alternative is common and constrained to be constant in all choice sets? With a constant alternative, respondents are not forced to make a choice among undesirable alternatives. Moreover, a constant alternative permits an estimate of demand volume rather than just market shares (Carson et al. 1994). A constant alternative can take many forms, ranging from the respondent's "current brand", to an indication that "none are acceptable", or simply "keep on shopping." While constant alternatives are often added to choice sets, little is

known about the efficiency implications of this practice. To create designs with a constant alternative, this alternative must be added to the candidate set. Also, a model with a constant alternative has one more parameter. Comparing a design with a constant alternative to one without, it is necessary to calculate D-error with respect to the original 32 parameters using the corresponding submatrix of Σ.

Adding a constant alternative to the original design increases the D-error of the original 32 parameters by 17% and is nearly 35% worse than allowing the fourth alternative to be variable. Some part of this loss in efficiency is due to the one additional degree of freedom from the constant alternative. A larger part is due to the efficiency lost when respondents are assumed to select the constant alternative. Every time it is chosen, one obtains less information about the values of the other parameters. In this case, the assumption that β=0 is not benign, as it assumes the constant alternative, along with all others in the four-option choice sets, will be chosen 25% of the time. We can reduce the efficiency cost to the other parameters by using a smaller β for the constant alternative, reflecting the assumption that it will be chosen less often.

In summary, the analysis suggests that adding a constant alternative to a three-alternative choice set *can* degrade the precision of estimates around the original parameters. Two caveats are important. First, this result will not always occur. We have found some highly fractionated designs where a constant alternative adds to the resolution of the original design. Second, there are studies where a major goal is the estimation of the constant alternative; in that case "oversampling" the constant ensures that its coefficient will be known with greater precision.

An important lesson across these four examples is that one cannot rely on heuristics to guide design strategies, ignoring statistical efficiency. It is generally necessary to test specific design strategies, given anticipated model parameters, to find a good choice design.

Evaluating Model Modifications. The proposed approach can be used to assess modifications of the model specification. This allows one, for example, to estimate the cost of "assumption insurance", i.e., building a design that is robust to false assumptions. Often we assume that factors are independent; for example, that the utility of price does not depend on brand or container. In many instances this assumption would be better termed a "presumption" in that if it is wrong, the estimates are biased, but there is no way to know given the design. Assessing the cost of assumption insurance involves four steps:

1. Find the best design for the unrestricted model (possibly including interactions).
2. Find the best design for the restricted model.
3. Evaluate D-error for that unrestricted design under the restricted model.
4. Evaluate D-error for the best design for the restricted model.

The cost of assumption insurance is the percent difference between steps 3 and 4, reflecting the loss of efficiency of the core parameters for the two designs. We illustrate how to assess this cost for a design that permits the price term to interact with brand and container versus one that assumes they are independent. To simplify the example, we take the same case as before, but assume that price is a linear rather than a categorical variable.[6]

[6] Substituting a linear price term for a three-level categorical one has two immediate implications. First, any change in coding results in quite different absolute values of D-error. Second, in optimizing a linear coding for price, the

The first step involves finding an efficient design with all price interactions with brand and brand×container estimable. This unrestricted model has 7 df for main effects (two for brand, two for container, two for flavor, and one for price), 12 df for two-way interactions (brand×price, brand×flavor, brand×container, container×price), and 4 df for the three-way interaction (brand×container×price). An efficient design for this unrestricted model has a D-error of .148. If this design is used for a restricted model in which price does not interact (7 df for main effects and 8 df for two-way interactions) then D-error drops to .118. The critical question is how much better still can one do by searching for the best design in the 15-parameter restricted model. The best design we find has a D-error of .110. Thus, assumption insurance in this case imposes a 6% (1-.110/.118) efficiency loss, a reasonable cost given that prices will often interact with brands and containers.

To summarize, the search routine allows estimates of the cost in efficiency of various design modifications and even changes in the model specification. Again, the important lesson is not the generalizations from the results of these particular examples, but rather an understanding of how these and similar questions can be answered in the context of any research study.

How Good Are These Designs?

The preceding discussion has shown that our adaptation of the modified Federov algorithm can find estimable choice designs and answer a variety of useful questions. We still need to discuss the question, how close to optimal are these designs? The search is nonexhaustive, and there is no guarantee that the solutions are optimal or even nearly so. For some designs, such as the alternative-specific one shown above, we can never be completely certain that the search process does not consistently find poor local optima. However, one can achieve some confidence from the pattern of results based on different random restarts; similar efficiencies emerging from different random starts indicate robustness of the resultant designs. An even stronger test is to assess efficiencies of the search process in cases where an optimal solution is known. While this cannot be done generally, we can test the absolute efficiency of certain symmetric designs, where the optimal design can be built using formal methods. We illustrated this kind of design in the three attribute, three level, three alternative, nine choice set (3^3/3/9) design discussed earlier, and found that the search routine was not able to find a better design. Now, we ask how good are our generated designs relative to three optimal designs: the design mentioned above and two corresponding, but bigger designs—a 4^4/4/16 and a 5^5/5/25 generic design.

For these types of designs we apply the proposed algorithm and compare our designs with the analytically generated ones. For each design, we used ten different random starts and three internal iterations. Figure 2 displays the impact of efficiency on different starting points and different numbers of internal iterations.

---

search routine will try to eliminate alternatives with the middle level of price within brand. This focus on extremes is appropriate given the linear assumption, but, may preclude estimation of quadratic effects.

Figure 2
CONVERGENCE PATTERN FROM DIFFERENT RANDOM STARTS

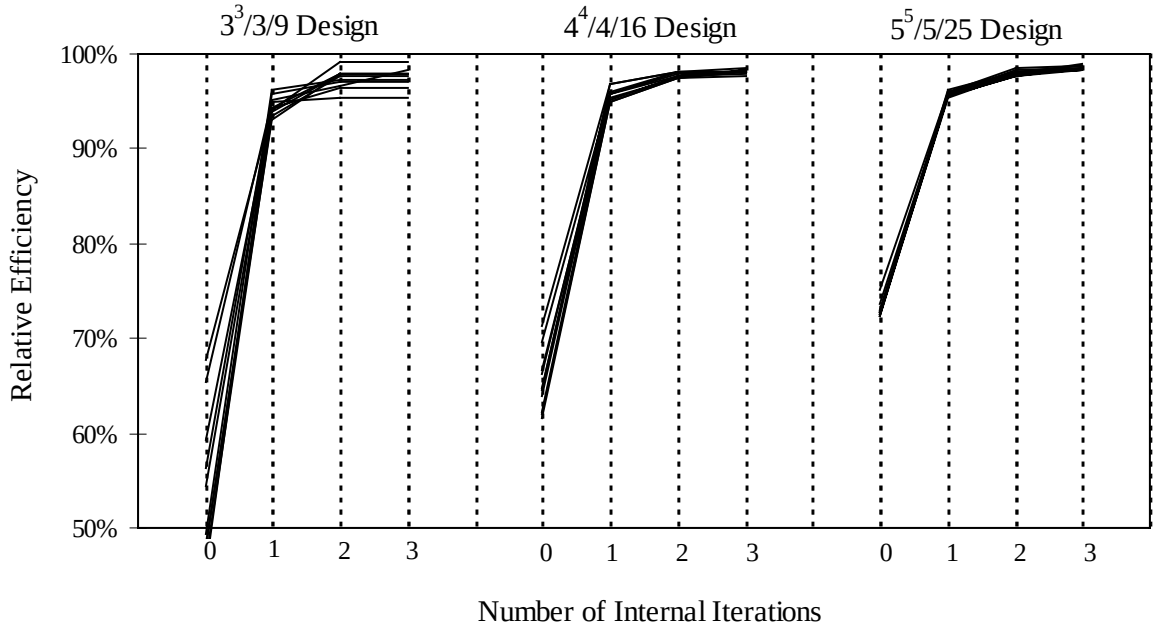


Figure 2 reveals important properties of the proposed algorithm. After the first iteration, the algorithm finds a choice design with about 90% relative efficiency, after a few more iterations, relative efficiencies approach 95%-99%. Further, this property appears to be independent of any initial starting design—the process converges just as quickly from a random start as from a rational one. These encouraging properties suggest important advantages for the practical use of the approach. First, in contrast to Huber and Zwerina (1996), the process does not require a rational starting design (which may be difficult to build). Second, since the process yields very efficient designs after only one or two iterations, most practical problems involving even large choice designs can be accommodated.

CONCLUSIONS

We propose an adaptation of the modified Federov algorithm to construct statistically efficient choice designs with standard personal computers. The algorithm generates efficient designs quickly and is appropriate for all but the largest choice designs. The approach is illustrated with a SAS/IML program. The SAS System has the advantage of a general model statement that facilitates the building of choice designs with different model specifications. The cost of using SAS/IML software, however, is that the algorithm generally runs slower than a program developed in, for example, PASCAL or C.

There are three major advantages of using a computer to construct choice designs rather than deriving them from formal principles. First, computers are the only way we know to build designs that allow one to incorporate anticipated model parameters. Since the incorporation of

this information can increase efficiency by 10% to 50% (see Huber and Zwerina 1996), this benefit alone justifies the use of computer search routines to find efficient choice designs.

The second advantage is that one is less restricted in design selection. Symmetric designs may not reflect the typically asymmetric characteristics of the real market. The adaptability of computerized searches is particularly important in choice studies that simulate consumer choice in a real market (Carson et al. 1994). Moreover, the process we propose allows the analyst to generate choice designs that account for any set of interactions, or alternative-specific effects of interest and critical tests of these assumptions. We illustrated a market emulation design that permits brand to interact with price, container, and flavor and can test the three-way interaction of brand by container by price. This pattern of alternative-specific effects would be very hard to build with standard designs, but it is easy to do with the computerized search routine by simply setting the model statement. The process can handle even more complex models, such as availability and attribute cross effects models (Lazari and Anderson 1994).

Finally, the ability to assess expected errors in parameters permits the researcher to examine the impact of different modifications in a given design, such as adding more choice sets or dropping a level from a factor. Most valuable, perhaps, is the ability to easily test designs for robustness. We provide one example of assumption insurance, but others are straightforward to generate. What happens to the efficiency of a design if there are interactions, but they are not included in the model statement? What kind of model will do a good job given a linear representation of price, but will also permit a test of curvature? What happens to the efficiency of the design if one's estimate of β is wrong?

There are several areas in which future research is needed. The first of these involves studies of the search process *per se*. We chose the modified Federov algorithm because it is robust and runs fast enough on today's desktop computers. As computing power increases, more exhaustive searches should be evaluated. For extremely large problems, faster and less reliable algorithms may be appropriate. Furthermore, while the approach builds efficient choice designs for multinomial logit models, efficiency issues with respect to other models, for example, nested logit and probit models, have yet to be explored.

A second area in which research would be fruitful involves the behavioral impact of different choice designs. The evaluations of our designs all implicitly assumed that the error level is constant regardless of the design. Many choice experiments use relatively small set sizes and few attributes reflecting an implicit recognition that "better" information comes from making the choice less complex. However, from a statistical perspective it is easy to show that smaller set sizes reduce statistical efficiency. In one example, we demonstrated that increasing the number of alternatives per choice set from three to four can increase efficiency by 16%. This gain depends on the assumption that respondent's error levels do not change. If they do increase, then that 16% percent gain might be lessened or even reversed. Thus, there is a need for a series of studies measuring respondents' error levels to tasks at different levels of complexity. Also, it is important to measure the degree of correspondence between the experimental tasks and the actual market behavior, choice experiments are intended to simulate. Such information is critical for correct trade-offs between design efficiency, measured here, and survey effectiveness, measured in the marketplace.

The purpose of this article is to demonstrate the important advantages of a flexible computerized search in generating efficient choice designs. The proposed adaptation of the modified Federov algorithm solves many of the practical problems involved in building choice

designs, thus enabling more researchers to conduct choice experiments. Nevertheless, we want to emphasize that it does not preclude traditional design skills; they remain critical in determining the model specification and in assessing the choice designs produced by the computerized search.

Appendix
SAS/IML CODE FOR THE PROPOSED CHOICE DESIGN ALGORITHM

The SAS code shows a simple implementation of the algorithm. In this example, the program finds a design with 27 choice sets and three alternatives per set. There are four attributes (brand, price, container, and flavor) each with three levels. A design is requested in which all main effects, the two-way interactions between brand and the other attributes, the two-way interaction between container and price, and the brand by price by container three-way interaction are estimable. Here, the parameters are assumed to be zero, but could be easily changed by setting other values.

A computer that evaluated all possible ($81^{81}/3!\cdot 27! = 5.9\cdot 10^{125}$) designs would take numerous billion years. Instead, we use the modified Federov algorithm, which uses the following heuristic: find the best exchange for each design point given all of the other candidate points. With 81 candidate alternatives, 27 choice sets, 3 alternatives per set, (say) 3 internal iterations, and 2 random starts, 81×27×3×3×2=39,366 exchanges must be evaluated. The algorithm tries to maximize $|\mathbf{X'X}|$ rather than minimizing $|(\mathbf{X'X})^{-1}|$ (note that $|(\mathbf{X'X})^{-1}|=|\mathbf{X'X}|^{-1}$). Each exchange requires then the evaluation of a matrix determinant, $|\mathbf{X'X}|$. Fortunately, we do not have to evaluate this determinant from scratch for each exchange since $|\mathbf{X'X} + \mathbf{x'x}| = |\mathbf{X'X}|\cdot|\mathbf{I} + \mathbf{x}\,(\mathbf{X'X})^{-1}\mathbf{x'}|$ (Mardia et al. 1979). Each exchange evaluates a quadratic form, and in this example with three alternatives per choice set, the determinant of a 3×3 matrix. It should also be noted that this algorithm can handle a rank-deficient covariance matrix by operating on $|\mathbf{X'X} + \mathbf{I}\cdot\varepsilon|$, where ε is a small number. This eliminates zero determinants so that less-than-full-rank codings and singular starting designs are not a problem. With these short cuts, one iteration required about 30 seconds on an ordinary 486 PC, implying that the algorithm is reasonable for many marketing contexts.

This appendix is provided simply to show the algorithm for those who might wish to implement or better understand it. If you want to use the algorithm, use the %CHOICEFF autocall SAS macro. This macro is described in more detail in the report “Multinomial Logit, Discrete Choice Modeling” (Kuhfeld, 2000). It is much larger and full-featured than the code shown in this appendix.

```
/*------------------------------Initial Set Up------------------------------*/
%let beta  = 0 0 0 0 0 0 0 0              /* 8 main effects                    */
             0 0 0 0 0 0 0 0              /* brand x price, brand x container,*/
             0 0 0 0                      /* brand x flavor,                   */
             0 0 0 0                      /* price x container interactions    */
             0 0 0 0 0 0 0 0;             /* brand x price x container         */
%let nalts = 3;                           /* Number of alternatives            */
%let nsets = 27;                          /* Number of choice sets             */

proc plan ordered;                        /* Create candidate alternatives     */
   factors brand=3 price=3 contain=3 flavor=3 / noprint;
   output out=candidat;
   run;

proc transreg design data=candidat;       /* Code the candidate alternatives   */
model class(brand price contain flavor brand*price brand*contain
            brand*flavor contain*price brand*contain*price / effects);
   output out=tmp_cand;
   run;

proc contents p data=tmp_cand(keep=&_trgind); run;

/*------------------------Begin Efficient Design Search----------------------*/
proc iml; file log;
   use tmp_cand(keep=&_trgind);           /* Identify candidate set for input */
   read all into cand;                    /* Read candidate set into IML      */
   utils  = exp(cand * {&beta}`);         /* exp(alternative utilities)       */
   np     = 1 / ncol(cand);               /* Exponent applied to determinant  */
   imat   = i(&nalts);                    /* Identity matrix                  */
   nobs   = &nsets # &nalts;              /* Total n of alts in choice design */
   ncands = nrow(cand);                   /* Number of candidates             */
   fuzz   = i(ncol(cand)) # 1e-8;         /* X`X ridge factor, avoid singular */

   start center(x, exputil);              /* Probability centering subroutine */
      do i = 1 to nrow(x) / &nalts;       /* Do for each choice set           */
         k = (i-1)#&nalts+1 : i#&nalts;   /* Choice set index vector          */
         p = exputil[k,]; p = p / sum(p); /* Probability of choice            */
         z = x[k,];                       /* Get choice set                   */
         x[k,] = (z - j(&nalts,1,1) *     /* Center choice set, absorb p's    */
                  p` * z) # sqrt(p);
      end;
   finish;

   /*---------------Create Designs With Different Random Starts---------------*/
   do desnum = 1 to 2;                    /* Number of designs to create      */

      indvec  = ceil(ncands *             /* Random index vector (indvec)     */
                uniform(j(1, nobs, 0)));  /* into candidates                  */
      des     = cand[indvec,];            /* Initial random design            */
      run center(des, utils[indvec,]);    /* Probability center               */
      currdet = det(des` * des);          /* Initial determinants, eff's      */
      maxdet  = currdet; oldeff = currdet ## np; fineff = oldeff;
      if fineff <= 0 then err = .; else err = 1 / fineff;
      put /// +8 'Design   Iteration  D-Efficiency      D-Error' /
              +8 '-----------------------------------------------';
      put +6 desnum 6. 0 10. +6 fineff best12. +2 err best12.;

      /*-------------------------Internal Iterations-------------------------*/
      do iter = 1 to 8 until(converge);   /* Iterate until convergence        */

         /*---------Consider Replacing Each Alternative in the Design---------*/
         do desi = 1 to nobs;             /* Process each alt in design       */
            ind = ceil(desi / &nalts);    /* Choice set number                */
            ind = (ind - 1) # &nalts + 1  /* Choice set index vector          */
                  : ind # &nalts;
            besttry = des[ind,];          /* Store current choice set         */
```

```
            des[ind,] = 0;                /* Remove current choice set        */
            do i = 0 to 100 until(d ## np > 1e-8);
               xpx = des`*des + i#i*fuzz; /* X`X, ridged if necessary         */
               d   = det(xpx);            /* Determinant, if 0 then X`X will  */
               end;                       /* be ridged to make it nonsingular */
            xpxinv = inv(xpx);            /* Inverse (all but current set)    */
            indcan = indvec[,ind];        /* Indvec for this choice set       */
            alt = mod(desi-1, &nalts) + 1;/* Alternative number               */

            /*-----------------Loop Over All of the Candidates----------------*/
            do candi = 1 to ncands;       /* Consider each candidate          */
               indcan[,alt] = candi;      /* Update indvec for this candidate */
               tryit = cand[indcan,];     /* Candidate choice set             */
               run center(tryit,          /* Probability center               */
                          utils[indcan,]);
               currdet = d *              /* Update determinant               */
                         det(imat + tryit * xpxinv * tryit`);

               /*------------Store Results When Efficiency Improves-----------*/
               if currdet > maxdet then do;
                  maxdet        = currdet;/* Best determinant so far          */
                  indvec[,desi] = candi;  /* Indvec of best design so far     */
                  besttry       = tryit;  /* Best choice set so far           */
                  end;
               end;

            des[ind,] = besttry;          /* Update design with new choice set*/
            end;

         /*----------Evaluate Efficiency/Convergence, Report Results----------*/
         neweff = maxdet ## np;           /* Newest efficiency                */
         converge = ((neweff - oldeff) /  /* Less than 1/2 percent            */
               max(oldeff,1e-8) < 0.005); /* improvement means convergence    */
         oldeff = neweff;                 /* Store for use in next iteration  */
         fineff = det(des` * des) ## np;  /* Efficiency at end of iteration   */
         if fineff <= 0 then err = .; else err = 1 / fineff;
         put +12 iter 10. +6 fineff best12. +2 err best12.;
         end;

      /*----Store Efficiency, Index of Efficient Design, Covariance Matrix----*/
      final = final // (shape(desnum || fineff, nobs, 2) || indvec`);
      cov = cov // (shape(desnum || fineff, ncol(des), 2) ||
                  sweep(des` * des, 1 : ncol(des)));
      end;

   /*----------------------Write Results to SAS Data Sets---------------------*/
   create cov     var({design effic &_trgind});   append from cov;
   create results var({design effic index   });   append from final;
   quit;

/*-------------Store Actual Design Points, Using Indices from IML-------------*/
data results; set results; i=index; n=_n_; set candidat point=i; run;
proc sort; by descending effic n; run;    /* Put most eff design first        */
proc print; run;                          /* Print designs, best to worst     */
```

REFERENCES

Addelman, Sidney (1962), "Symmetrical and Asymmetrical Fractional Factorial Plans," *Technometrics*, 4, 47-58.

Anderson, Don A. and James B. Wiley (1992), "Efficient Choice Set Designs for Estimating Cross-Effects Models," *Marketing Letters, 3,* 357-70.

Bunch, David S., Jordan J. Louviere, and Donald A. Anderson (1996), "A Comparison of Experimental Design Strategies for Choice-Based Conjoint Analysis with Generic-Attribute Multinomial Logit Models," *Working Paper*, Graduate School of Management, University of California at Davis.

Carson, Richard T., Jordan J. Louviere, Don A. Anderson, Phipps Arabie, David S. Bunch, David A. Hensher, Richard M. Johnson, Warren F. Kuhfeld, Dan Steinberg, Joffrey Swait, Harry Timmermans, and James B. Wiley (1994), "Experimental Analysis of Choice," *Marketing Letters*, 5.4 (October), 351-67.

Cook, R. Dennis and Christopher J. Nachtsheim (1980), "A Comparison of Algorithms for Constructing Exact D-optimal Designs," *Technometrics*, 22, 315-24.

Federov, Valery V. (1972), *Theory of Optimal Experiments*, translated and edited by W. J. Studden and E. M. Klimko, New York: Academic Press.

Green, P. E. (1974), "On the Design of Choice Experiments involving Multifactor Alternatives," *Journal of Consumer Research*, 1, 61-8.

Huber, Joel, and Klaus Zwerina (1996), "The Importance of Utility Balance in Efficient Choice Designs," *Journal of Marketing Research*, 33 (August), 307-17.

Krieger, Abba B. and Paul E. Green (1991), "Designing Pareto Optimal Stimuli for Multiattribute Choice Experiments," *Marketing Letters*, 2, 337-48.

Kuhfeld, Warren F. (2000), "Multinomial Logit, Discrete Choice Modeling," Marketing Research Methods in the SAS System, Author.

_____, Randall D. Tobias, and Mark Garratt (1994), "Efficient Experimental Design with Marketing Research Applications, " *Journal of Marketing Research*, 31 (November), 545-57.

Lazari, Andreas G. and Don A. Anderson (1994), "Designs of Discrete Choice Experiments for Estimating Both Attribute and Availability Cross Effects," *Journal of Marketing Research*, 31 (August), 375-83.

Louviere, Jordan J. and George Woodworth (1983), "Design and Analysis of Simulated Consumer Choice of Allocation Experiments: A Method Based on Aggregate Data," *Journal of Marketing Research,* 20 (November), 350-67.

Mardia, Kanti V., John T. Kent, and John M. Bibby (1979), *Multivariate Analysis*, New York: Academic Press.

McFadden, Daniel (1974), "Conditional Logit Analysis of Qualitative Choice Behavior," in *Frontiers in Econometrics*, P. Zarembka, ed., New York: Academic Press, 105-42.

Raktoe, B. L., A. Hedayat, and W.T. Federer (1981), *Factorial Designs*, New York: John Wiley & Sons.