Acta Univ. Sapientiae, Economics and Business, **2,** (2014) 95–108

DOI: 10.2478/auseb-2014-0012

A Comparison of Algorithms for Conjoint Choice Designs

László ILLYÉS
Sapientia Hungarian University of Transylvania, Faculty of Economic and Human Sciences, Department of Economic Sciences, Miercurea Ciuc
E-mail: illyeslaszlo@sapientia.siculorum.ro

Zsolt SÁNDOR
Sapientia Hungarian University of Transylvania, Faculty of Economic and Human Sciences, Department of Business Sciences, Miercurea Ciuc
E-mail: sandorzsolt@sapientia.siculorum.ro

Abstract. We study the performance of some widely-used design construction algorithms like the coordinate exchange and swapping-cycling as well as some of their versions. We measure performance in two ways, namely, first, by measuring jointly both running time and the efficiency achieved, and, second, by fixing the running time of the algorithms and measuring the efficiency achieved while allowing the number of choice situations to vary. In addition, we also analyse the performance in terms of heterogeneous designs. A somewhat surprising outcome of our analyses is that a simplified version of the joint swapping-cycling algorithm outperforms the coordinate-exchange algorithm irrespective of the performance measure.

Keywords and phrases: survey, logit model, greedy approach.
JEL classification: C15, C83

1. Introduction

Choice experiments are widely used to study consumer preferences for various products in different areas like marketing, transportation, environmental and health economics. Their popularity stems from the fact that they can be used to elicit preferences for hypothetical products. An important question in designing a choice experiment is what hypothetical products to present to the respondents who participate in the experiment. One approach in the literature was to design choice experiments based on statistical efficiency.

Conjoint choice designs based on statistical efficiency are typically constructed by optimizing an objective function that is a scalar function of the information

matrix of the model considered (Huber & Zwerina, 1996; Sándor & Wedel, 2001). The variables in this optimization are the attributes of the hypothetical products, which are taken to be discrete. This implies a discrete optimization problem, whose complexity depends on the number of choice situations involved in the experiment and on the complexity of the underlying model.

The need for considering complex models stems from the need for modelling consumer preferences more realistically. Examples from this line of literature are Sándor and Wedel (2002) and Bliemer and Rose (2010), who use the random coefficient logit to model consumer heterogeneity in preferences for attributes. While the design construction problem is already rather complex in the former, the latter paper considers the panel version of the random coefficient logit, which allows for correlation between the choices made by a given consumer in different choice situations. The computational complexity of evaluating the objective function for this model is much more severe than in previously considered models. Moreover, if, for this model, we allow for a much larger number of choice situations, like in the heterogeneous design considered by Sándor and Wedel (2005), then the computational complexity becomes even more severe.

Therefore, we believe that the performance of the design construction algorithms in terms of speed is crucial. In the literature, several different algorithms have been proposed. Meyer and Nachtsheim (1995) proposed the coordinate-exchange algorithm, which has the appealing feature of being the simplest algorithm for the problem. Huber and Zwerina (1996) proposed the relabeling and swapping algorithms, while Sándor and Wedel (2001) proposed the cycling algorithm. The three latter algorithms have been used jointly or in pairs (relabeling-swapping or swapping-cycling) in order to attain higher efficiency.

In this paper, we study the performance of these algorithms and some of their versions. We measure performance in two ways, namely, first, by measuring jointly both running time and the efficiency achieved and, second, by fixing the running time of the algorithms and measuring the efficiency achieved while allowing the number of choice situations to vary. In addition, we also analyse the performance in terms of heterogeneous designs. A somewhat surprising outcome of our analyses is that a simplified version of the joint swapping-cycling algorithm outperforms the coordinate-exchange algorithm irrespective of the performance measure.

The remainder of the paper is organized as follows. In Section 2, we present the conjoint choice design problem in the context of the logit model. Section 3 explains the algorithms used. Section 4 presents results from comparing the algorithms for homogeneous and heterogeneous designs. In the last section, we conclude and outline further directions of research.

2. Conjoint Choice Designs for the Logit Model

In a choice experiment, a respondent is presented a conjoint choice design that consists of several choice sets of hypothetical products characterized by their attributes. The respondent is supposed to choose the best alternative in each choice set. The choice data collected in this way can be used to estimate the parameters of the underlying model. Pioneering work in conjoint choice design (e.g., Huber & Zwerina, 1996) assumed the logit model. In this paper, we use this model for two reasons. First, we provide a comparison of the different algorithms for the most commonly used model in the literature. Second, we provide a set of results that can be used as reference for studying the performance of different algorithms applied for more complicated models.

In the logit model, the utility of respondent $i = 1,...,N$ for hypothetical product $j = 1,...,J$ in choice set $s = 1,...S$ is specified as:

$$U_{ijs} = x'_{ijs}\beta + \varepsilon_{ijs}, \tag{1}$$

x_{ijs} is a k x1-vector of attributes of alternative j, β is a k x1-vector of parameters weighting these attributes, and ε_{ijs} is an error term having an i.i.d. type I extreme value distribution. The probability that profile j is chosen from the choice set s has the closed form

$$p_{ijs} = \frac{\exp(x'_{ijs}\beta)}{\sum_{r=1}^{J}\exp(x'_{irs}\beta)}. \tag{2}$$

The information matrix can be computed as the variance of the first-order derivatives of the log-likelihood function, and it is equal to the sum of the choice-set specific information matrices:

$$I(\beta) = \sum_{i=1}^{N}\sum_{s=1}^{S} X'_{is}(P_{is} - p_{is}p'_{is})X_{is}, \tag{3}$$

where $X_{is} = [x_{i1s},...,x_{iJs}]'$ is the matrix of the attributes in choice set s, $p_{is} = [p_{i1s},...,p_{iJs}]'$ and $P_{is} = diag(p_{i1s},...,p_{iJs})$. In order to study the algorithms with different objective functions, we consider two scalar transformations of the information matrix, namely the $D - error = \det[I(\beta)]^{-1/k}$, which uses the determinant, and the $A - error = tr(I^{-1}(X,\beta))$, which uses the trace of the inverse of the information matrix (Kessels et al., 2006). Both design criteria should be minimized with respect to the elements of X_{is} for all consumers i and choice sets s. This approach corresponds to the so-called heterogeneous design approach; when the designs presented to respondents are the same, that is $X_{is}=X_s$ for all i and s, and then we talk about the homogeneous design approach. We note that the design criteria depend on the parameters that need to be estimated; in order to

deal with this issue, we follow the locally optimal design approach by assuming a certain value for these parameters when constructing the design.

In this paper, we consider designs with 15 choice sets with 2 alternatives in each. The alternatives are assumed to be hypothetical products having 4 attributes with 3 levels denoted by 1, 2 and 3. The attributes are qualitative variables coded as [1 0], [0 1] and [-1 -1]. The assumed value of the parameter-vector is the transposed of [-1, 0, -1, 0, -1, 0, -1, 0].

3. Algorithms Used

First, we present the core algorithms, and then we specify the versions and combinations that we use. We illustrate each algorithm by means of the choice set example from *Table 1*.

Table 1. *Example of a choice set*

	Attribute 1	**Attribute 2**	**Attribute 3**	**Attribute 4**
Alternative 1	3	2	3	2
Alternative 2	2	1	2	1

Swapping (Huber & Zwerina, 1996) changes the level values between the two alternatives for the same attribute. In the example (see *Table 2*) values, 3 from the first alternative is changed with value 2 from the second alternative. It starts with the first attribute of the first choice set and continues with the second, third and fourth attributes, and then it proceeds in a similar way with the second choice set until the last choice set.

Table 2. *Swapping the 1st attribute*

	Attribute 1	**Attribute 2**	**Attribute 3**	**Attribute 4**
Alternative 1	2	2	3	2
Alternative 2	3	1	2	1

Cycling (Sándor & Wedel, 2001) changes both levels of the two alternatives corresponding to the same attribute in the following manner: 1→2, 2→3 and 3→1. In the example (Table 3), cycling is applied to the levels of the first attribute; so, 3 becomes 1 and 2 becomes 3. It starts with the first attribute of the first choice set and continues with the second, third and fourth attributes, and then it continues with the second, third etc. choice sets in a similar fashion.

Table 3. *Cycling the original levels for the 1st attribute*

	Attribute 1	Attribute 2	Attribute 3	Attribute 4
Alternative 1	1	2	3	2
Alternative 2	3	1	2	1

The coordinate-exchange algorithm (Meyer & Nachtsheim, 1995) changes one attribute level at a time to the other values. In the example (Table 4), the level of the first attribute of the first alternative is changed from 3 to 1 and then to 2. It starts with the first attribute of the first alternative in the first choice set; then, it continues with the levels of the second, third and fourth attributes, and then it proceeds in a similar way with the attributes of the second alternative and the other choice sets.

Table 4. *Coordinate exchange for the 1st alternative in the 1st attribute*

	Attribute 1	**Attribute 2**	**Attribute 3**	**Attribute 4**
Alternative 1	1	2	3	2
Alternative 2	2	1	2	1
Alternative 1	2	2	3	2
Alternative 2	2	1	2	1

For each algorithm, those changes that improve on the design criterion are preserved and those that do not improve are discarded. The original swapping (henceforth sw) and cycling (henceforth cy) algorithms restart from the beginning after each improving change. We have found that this feature slows down the algorithms considerably; so, we have introduced versions of these algorithms that do not restart after improving changes, but only after reaching the last attribute of the last choice set. We refer to these versions as the *no restart* versions, and we denote it by *nr*. The original coordinate-exchange algorithm does not restart from the beginning after improving changes; so, it is of *nr*-type by default; we now consider a version that restarts from the beginning after each improving change. We denote the latter algorithm by *kx* and the former by *kxnj*. Our proposed algorithms for evaluation are presented in *Table 5*.

Table 5. *The algorithms*

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
sw_cy	*swnj_cynj*	*swnj_cy*	*sw_c.nr*	*cynj*	*swnj*	*kxnj*	*kx*

The first algorithm (*sw_cy*) is the one used by Sándor and Wedel (2001) without the relabeling algorithm. Swapping is applied first and, when this does not give any improvement, then the algorithm proceeds further by cycling. The second algorithm uses swapping first without restart, and then cycling without restart. The third algorithm uses swapping without restart and the original cycling. The fourth algorithm first runs the original swapping, and then the no restart cycling. The 5th and 6th algorithms are pure cycling and swapping with no restart, while the 7th is the original coordinate exchange and the 8th is the coordinate exchange with restart. The original coordinate-exchange algorithm is remarkable in that it can be regarded as the simplest algorithm conceptually.

4. Results

We present two sets of results: one for homogeneous and one for heterogeneous designs.

Homogeneous designs

For each algorithm mentioned in *Table 5*, we construct 20 designs using the same 20 (level-balanced) starting designs. We measure the average running time of each algorithm and we represent the results in scatter plots of points, where the vertical axis shows the average running time in seconds for each algorithm. In one set of results, we analyse the average of the design criterion values, while in another set of results we look at the minimum of the design criterion values. The latter results aim at capturing the performance of the algorithms for finding a design that is closer to the globally optimal design.

Figure 1 presents the scatter plot in the case of the D-error criterion. We can observe that *sw_cy* and *swnj_cy* are the best algorithms on average in terms of the D-error, while *swnj* and *kxnj* are the best on average in terms of running time. By considering both measures jointly, we can notice that both *kx* and *sw_cynj* are worse than *swnj_cy* with respect to both measures, that is they produce less efficient designs and run longer on average.

Figure 2, which presents the scatter plot in the case of the A-error criterion, leads to qualitatively similar findings. Again, *sw_cy* and *swnj_cy* are the best algorithms on average in terms of the design criterion, while *swnj* and *kxnj* are the best on average in terms of running time. By considering both measures jointly, *kx* is worse again than *swnj_cynj* with respect to both measures. Similarly, *cynj* is worse than *swnj_cynj* with respect to both measures.

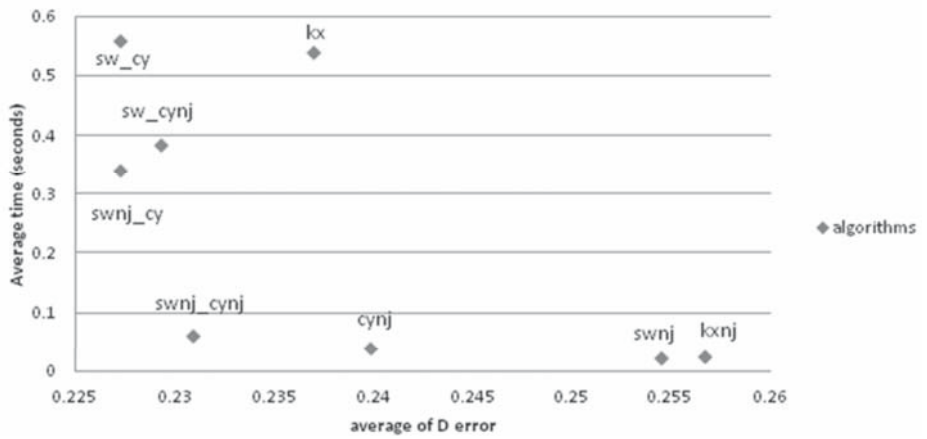

Figure 1. *Average running time and average D-error for homogeneous design*

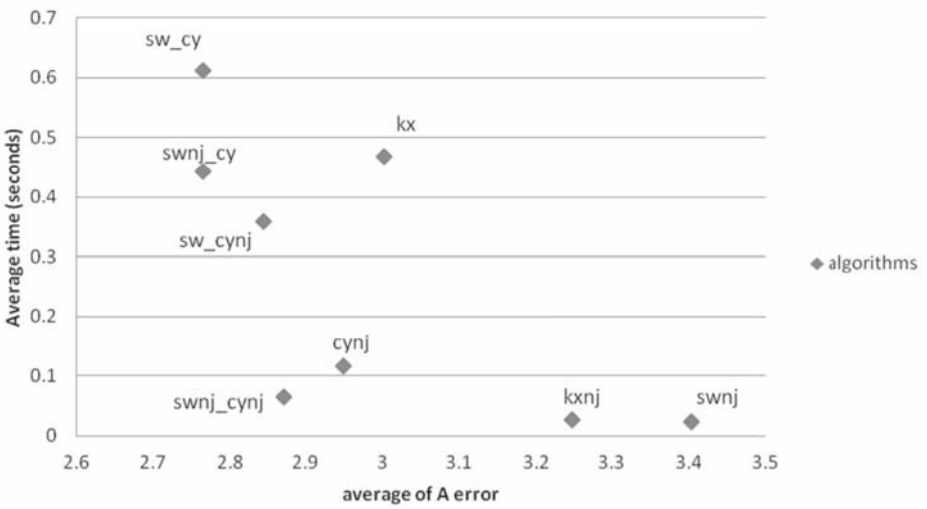

Figure 2. *Average running time and average A-error for homogeneous design*

Figures 3 and 4 present the scatter plots when instead of the average D-errors the minima of these are plotted on the horizontal axis. Qualitatively, *Figure 3* is very similar to *Figure 1* and *Figure 2* is very similar to *Figure 4*, and in fact similar conclusions can be drawn regarding the relative performance of the algorithms.

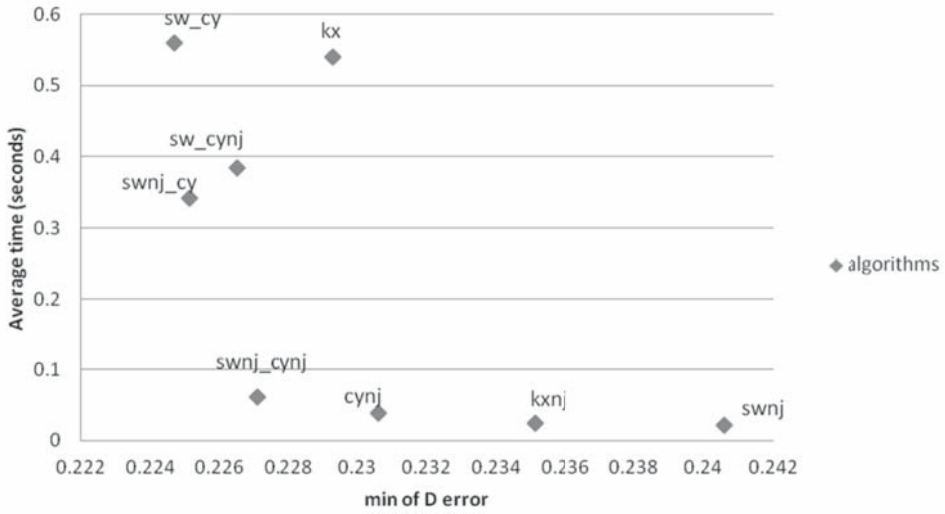

Figure 3. *Average running time and min of D-error for homogeneous design*

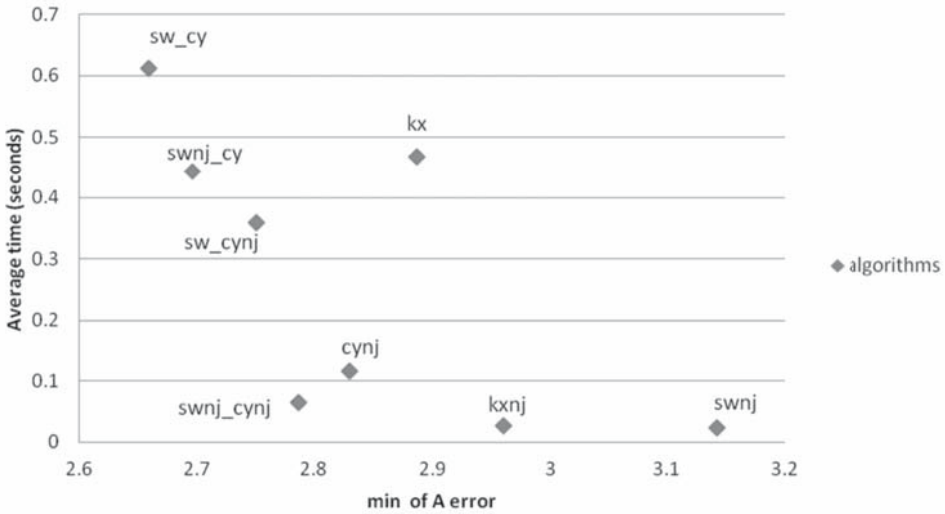

Figure 4. *Average running time and min of A-error for homogeneous design*

We intend to determine the algorithm(s) that has (have) the best performance taking into account figures 1–4. First, we note that since *kx*, *sw_cynj* and *cynj* are dominated in terms of both measures either in the case of the D-error or in the case of the A-error, they cannot have the best performance, so they can be discarded. Further, figures 1 and 3 suggest that *sw_cy*, *swnj_cy* and *swnj_cynj* are similar in terms of the D-error because the percentage difference between the worst and the best is less than about 2.2%, and it is known that this means

that a percentage increase of at most 2.2 % in the number of consumers used for the worst design is sufficient to match the performance of the best design. Out of these three algorithms, *swnj_cynj* needs the least running time; so, we discard the other two algorithms. Therefore, the best algorithm is one of the following: *swnj_cynj*, *kxnj* and *swnj*.

Since the running times of these algorithms are slightly different and they yield designs with different D-errors and A-errors, we compare them by fixing the running time to 120 seconds and running the algorithms in this time with different starting designs as many times as possible. The minima of the design criteria obtained are presented in *Table 6*. For example, for the fastest algorithm in the case of the D-error (*swnj*), we obtained 5,792 designs and the minimum of their D-errors is 0.24978. Also, the slowest algorithm (*sw_cy*), which is not presented in *Table 6*, in this case, produced 228 designs. For both design criteria, the best algorithm turns out to be *swnj_cynj*: its lowest value in the D-error case is 0.23063 and in the A-error case is 2.862. This is followed by *swnj* and *kxnj* in the case of the D-error criterion, while in the case of the A-error criterion by *kxnj* and *swnj*.

Table 6. *Minimum D- and A-errors for three selected algorithms with a running time of 120 seconds*

	swnj	**kxnj**	**swnj_cynj**
D1design	0.24978	0.25944	0.23063
D10des	0.02242	0.02262	0.02217
A1design	3.44375	3.30272	2.86219
A10des	0.27879	0.27320	0.26043

It is important to mention that the percentage difference in D-error between the designs obtained by the *kxnj* and *swnj_cynj* algorithms is 11.1%. On the one hand, this means that one needs by 11.1% more respondents when using the *kxnj* algorithm than when using the *swnj_cynj* algorithm. On the other hand, this difference means that the *kxnj* algorithm is more likely to get stuck at local optima than the *swnj_cynj* algorithm.

Heterogeneous Designs

A heterogeneous design is a design in which different respondents are given different designs. So, the main distinction with respect to homogeneous design is that in the latter respondents get the same design. The main motivation for using heterogeneous design, as shown by Sándor and Wedel (2005), is that it offers higher statistical efficiency with the same number of respondents since the design is optimized with fewer constraints. These authors also show that it

is not necessary that every respondent get a design different from the others; it is sufficient to use six different designs for all respondents.

First, we present an analysis of heterogeneous designs that is analogous to that presented in figures 1 and 2. *Figure 5* shows that *sw_cy* and *swnj_cy* are again the best algorithms on average in terms of the D-error and *swnj* and *kxnj* are the best on average in terms of running time. By considering both measures jointly, we can again notice that both *kx* and *sw_cynj* are worse than *swnj_cy* with respect to both measures.

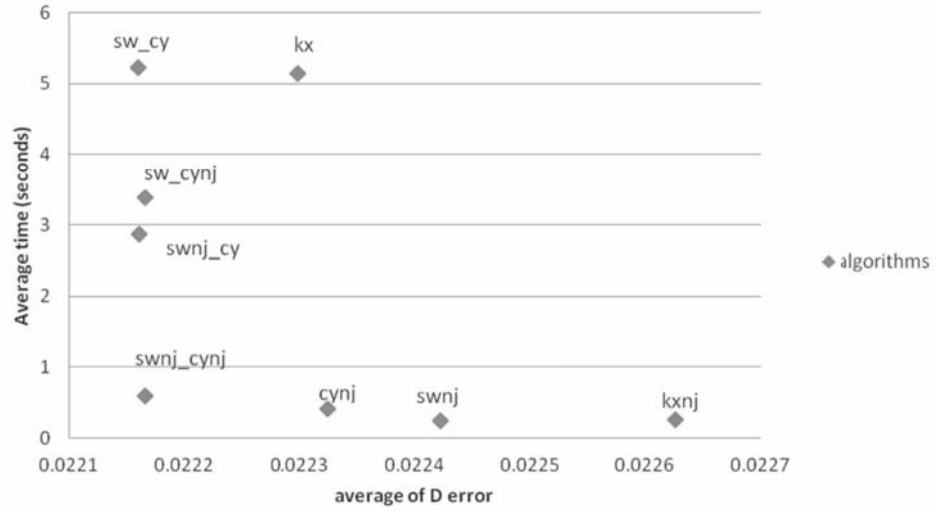

Figure 5. *Average running time and D-error for heterogeneous design*

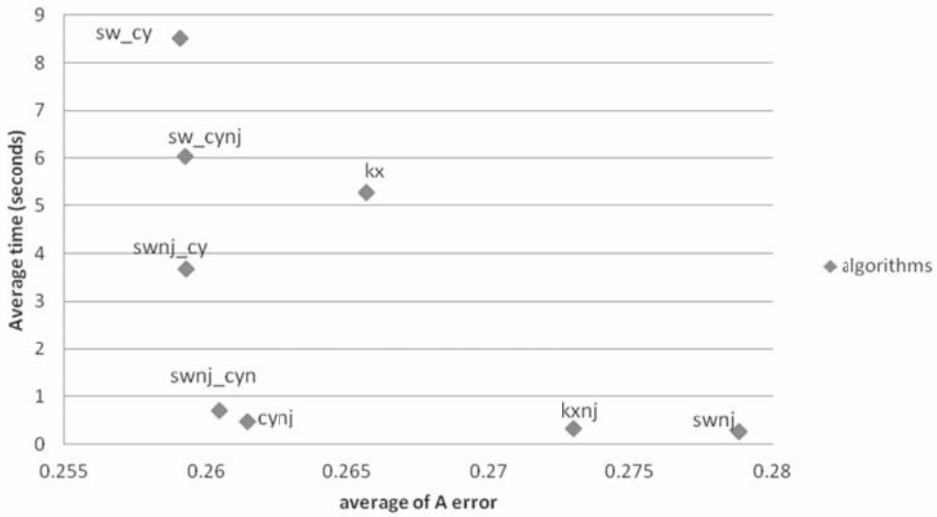

Figure 6. *Average running time and A-error for heterogeneous design*

Figure 6 shows that *sw_cy*, *swnj_cy* and *swnj_cy* are the best algorithms on average in terms of the A-error criterion, while *swnj* and *kxnj*, as in *Figure 2*, are again the best on average in terms of running time. By considering both measures jointly, *kx* is worse again than *swnj_cynj* with respect to both measures.

Figures 5 and 6 yield a conclusion that is qualitatively similar to the homogeneous design case. We intend to determine the algorithm(s) that has (have) the best performance taking into account figures 5-6. First we note that since *kx* and *sw_cynj* are dominated in terms of both measures either in the case of the D-error or in the case of the A-error, they cannot have the best performance, so we discard them. Further, figures 5 and 6 suggest that *sw_cy*, *swnj_cy* and *swnj_cynj* are rather similar in terms of both the D- and A-error. Out of these three algorithms, *swnj_cynj* needs the least running time; so, we discard the other two algorithms. Therefore, the best algorithm is one of the following: *swnj_cynj*, *cynj*, *kxnj* and *swnj*.

Similar to the homogeneous case, since the running times of these algorithms are slightly different and they yield designs with different D-errors and A-errors, we compare them by fixing the running time to 120 seconds and running the algorithms in this time with different starting designs as many times as possible. The minima of the design criteria obtained are presented in *Table 6*. For both design criteria, the best algorithm turns out to be again *swnj_cynj*: its lowest value in the D-error case is 0.02217 and in the A-error case is 0.26043.

The percentage difference in D-error between the designs obtained by the *kxnj* and *swnj_cynj* algorithms is 2%, which is clearly lower than in the homogeneous design case. This means that the difference between the *kxnj* and the *swnj_cynj* algorithms in terms of local versus global optimality is less pronounced in the heterogeneous design case.

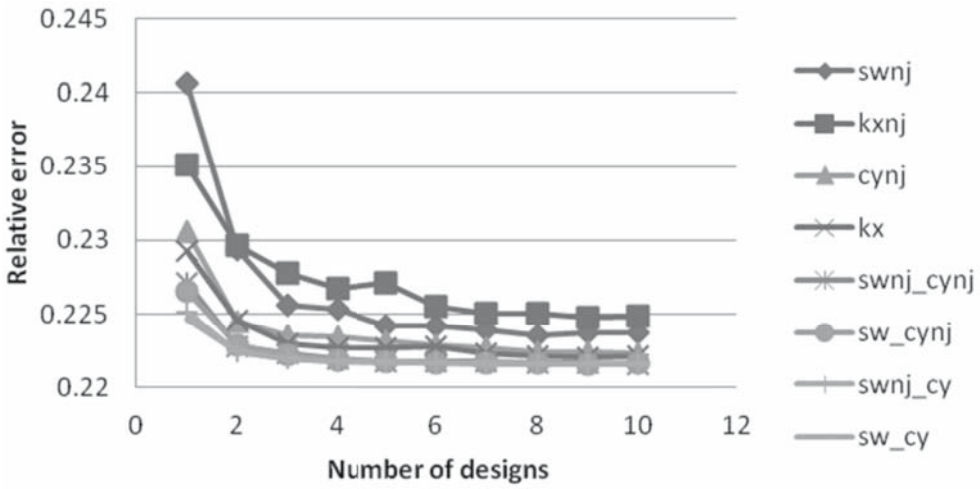

Figure 7. *D-errors relative to the number of different designs*

In order to present further insights regarding the algorithms in the heterogeneous design case, we present their performance based on 1, 2,… 10 different designs (figures 7 and 8). We refer to the design criteria as relative D- and A-error because we multiply these by the number of different designs used. This way, the relative design errors measure the marginal effect of using an additional different design.

The first impression from *Figure 7* is that for all the algorithms the marginal improvement in relative D-error diminishes as the number of designs increases. Further, for the algorithms *sw_cy*, *swnj_cy*, *sw_cynj* and *swnj_cynj*, the relative D-error becomes constant for 4-5 designs or more (a similar finding is reported in Sándor and Wedel, 2005). The algorithms *kx* and *cynj* come close to this constant, but only for 9-10 designs. The algorithms *swnj* and *kxnj* reach constant relative D-error values that are higher. This kind of behaviour of *swnj* is not surprising because swapping preserves the level balance property; so, this algorithm searches in a design space that is smaller than that searched by the other algorithms. The fact that *kxnj* reaches an even higher relative D-error constant is somewhat unexpected. We believe that it is related to the finding mentioned above that the coordinate-exchange algorithm seems to be more likely to get stuck in local optima than the other algorithms.

Figure 8 is in essence similar to *Figure 7*. We can, however, notice that the two coordinate-exchange algorithms and, to a lesser extent, the *swnj* algorithm do not display a monotonically decreasing trend. Again, we believe that this is related to the fact that the coordinate-exchange algorithm seems to be more likely to get stuck in local optima than the other algorithms.

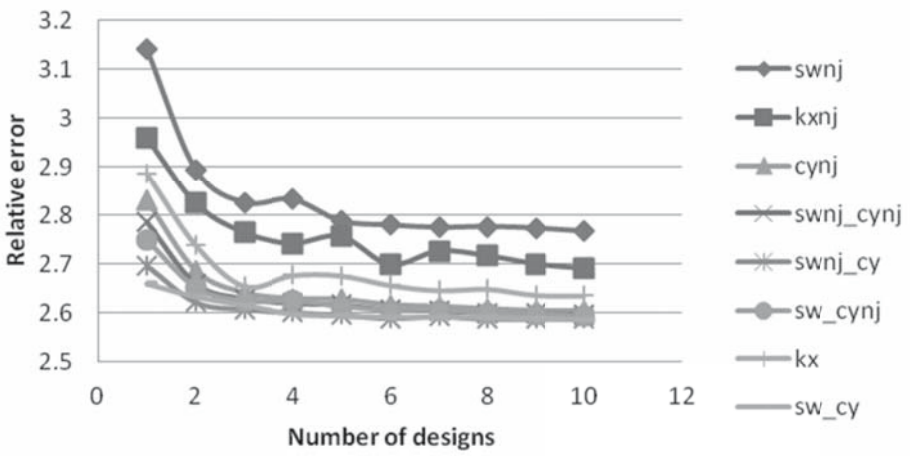

Figure 8. *A-errors relative to the number of different designs*

Finally, we mention that we implemented all the algorithms for heterogeneous designs as so-called greedy algorithms. That is, instead of optimizing all the designs

jointly, we have first optimized one design, then the second design only while keeping the first one fixed, then the third design while keeping the first and second designs fixed, and so on. For more details, we refer to Sándor and Wedel (2005).

5. Conclusions

This paper compares the swapping-cycling (Huber & Zwerina, 1996; Sándor & Wedel, 2001) and the coordinate-exchange (Meyer & Nachtsheim, 1995) algorithms regarding their speed and relative optimality. The comparisons include versions of the swapping and cycling algorithms that do not restart from the beginning of the design after each successful modification as well as a version of the coordinate-exchange algorithm that restarts after each successful modification. The comparisons are done for both homogeneous and heterogeneous designs.

The main outcome of our results is that the joint swapping-cycling algorithm without restart outperforms the other algorithms – thus, the coordinate-exchange algorithm as well – both in the homogeneous and heterogeneous design cases. An interesting implication of this finding is that the simplicity of the coordinate-exchange algorithm that is viewed as an appealing feature does not necessarily imply that the algorithm performs well with respect to speed and optimality. Our final conclusion is that researchers should use the joint swapping-cycling algorithm without restart when they adopt the logit for modelling consumer choice.

It would be interesting to know if the same conclusion can be reached in the case of more realistic models like random coefficient logit. Besides locally optimal design, Bayesian design, which – instead of assuming some values for the model parameters – assumes that their distribution is known, is another important design problem that may not lead to the same conclusion. The performance of some global optimization procedures (e.g. Genetic Algorithms, Tabu Search, Simulated Annealing) in conjunction with the design algorithms discussed in this paper may also change the conclusion of this paper. We intend to study these problems in the future.

Acknowledgements

This work was supported by a grant of the Romanian Ministry of National Education, CNCS – UEFISCDI, project number PN-II-ID-PCE-2012-4-0066.

References

Bliemer, Michiel C. J.; Rose, John M. (2010). Construction of experimental designs for mixed logit models allowing for correlation across choice observations, *Transportation Research B* 44: 720–734.

Huber, J.; Zwerina, K. (1996). The importance of utility balance in efficient choice designs, *Journal of Marketing Research* 33: 307–317.

Kessels, R.; Goos, P.; Vandebroek, M. (2006). A comparison of criteria to design efficient choice experiments, *Journal of Marketing Research* 43: 409–419.

Meyer, R. K.; Nachtsheim, C. J. (1995). The coordinate-exchange algorithm for constructing exact optimal experimental designs, *Technometrics* 37: 60–69.

Sándor, Z.; Wedel M. (2001). Designing conjoint choice experiments using managers' prior beliefs. *Journal of. Marketing Res.* 38(November): 430–444.

Sándor, Z.; Wedel M. (2002). Profile construction in experimental choice designs for mixed logit models, *Marketing Science* 21 (4): 55–75.

Sándor, Z.; Wedel, M. (2005). Heterogeneous conjoint choice designs, *Journal of Marketing Research* 42: 210–218.