Alex Graber

Andrew Armstrong

D-Optimality with Level Balance Constraints

**Abstract**

There is much research on the topic of identifying efficient experimental designs (Hauser & Rao, 2002) that require the fewest experiments to extract the most information. The current standard seems to be *D-error*, the geometric mean of the eigenvalues of the covariance matrix (*D-efficiency* is the inverse of *D-error*) (Kuhfeld, Huber, & Zwerina, 1996). Thus, the goal of an efficient design is to minimize *D-error* (thus maximizing *D-efficiency*). It has been shown that *D-efficient* designs satisfy four principles: orthogonality, level balance, minimal overlap, and utility balance. The Fedorov algorithm is an exhaustive search method developed in 1969, which optimizes the *D-efficiency* of a set of candidates with respect to the set of all possible candidates. Typically, the Fedorov Algorithm is unable to deal with constraints within the design matrix, such as constraints on level balance. Here, we have added constraints to the design of the algorithm and it gave an implementable solution. We have accomplished this through two methods: an analytical method, explicitly incorporating constraints into the problem, and a numerical method, using a genetic algorithm. We have found that though the proposed analytical method does converge, it runs into computation overhead problems with larger candidate sets. The genetic algorithm will identify approximate solutions, but as a stochastic search method, cannot guarantee the optimal constrained D-efficient design.

**Introduction:**

A pharmaceutical market research firm uses simulated patient treatment as a method to understand physician demand in specific treatment areas. In this method, a limited universe of patients is designed in order to represent as much of the actual disease area patient universe as possible. Patients are defined by multiple attributes (age, gender, BMI, etc.), and each attribute may have multiple levels (male/female, etc.). The breakdown of attribute levels is provided by a specified distribution to be achieved in the simulated universe. These simulated patients are then treated, where a given treatment (yes/no) can be related back to the patient design.

Patient design in this manner is a specialized choice methodology somewhat analogous to conjoint. In both conjoint and patient simulation, respondents are forced to make a decision based on a stimulus that is composed of multiple attributes and levels (Kuhfeld, Huber, & Zwerina, 1996). When the number of attributes and levels grow beyond a small set, presenting the full design (full factorial) becomes a challenge due to both the number of combinations required and the amount of burden placed on the respondent. Fractional factorial designs, then, seek to allow the research to eke as much data out of the analysis as possible but use a much more limited subset of stimuli.

Much research has been done on the topic of identifying efficient experimental designs (Hauser & Rao, 2002). The current standard seems to be *D-error* – roughly, the geometric mean of the eigenvalues of the covariance matrix (*D-efficiency* is the inverse of *D-error*) (Kuhfeld, Huber, & Zwerina, 1996). Thus, the goal of an efficient design is to minimize *D-error* (thus maximizing *D-efficiency*). It has been shown that *D-efficient* designs satisfy four principles: 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” (Kuhfeld, Huber, & Zwerina, 1996).

The standard method to identify an efficient design is to use one of any variant of the Fedorov Algorithm which, given a starting design, recursively makes exchange(s) that reduce *D-error* until some convergence criteria is met. This method is susceptible to local minima; it may be necessary to run multiple iterations of the Fedorov Algorithm with different random starting designs to find the most efficient design (Kuhfeld, Huber, & Zwerina, 1996).

**Initial Problem Description:**

Contrary to standard *D-optimal* designs, the design for patient simulation must allow *D-error* as a result of violating the principle of level balance. Since the goal is for the simulated patient universe to map to the actual patient universe, the researcher may need to control for the distribution of levels within each attribute. Additionally, certain attributes and levels may have required interactions (i.e., a patient must be female to be pregnant).

As a toy problem, let us consider a patient universe in which patients are defined by:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | Distribution |
| **Age** | Youth | Adult | Elderly | 25/50/25 |
| **Gender** | Male | Female |  | 50/50 |
| **BMI** | Underweight | Normal | Obese | 25/25/50 |

Expanding out all possibilities into the entire candidate set, this would be 3\*3\*2\*3 = 54 unique patient profiles. Given respondent time is expensive, and high respondent burden decreases quality of results, we seek to reduce time-in-survey by creating a fractional-factorial design of 8 unique patient profiles. As we want to extract as much data from the exercise as possible, the 8-profile fractional-factorial design must be as efficient as possible.

Practically speaking, the number of attributes is limited to no more than 25, each with at most 5 levels due to the complexity of the simulation, limited respondent pool, and limited number of experiments possible per respondent. Thus, at most, the candidate set contains 5^25 (approx. 3x1017) possibilities – and will generally be significantly smaller as not all 25 attributes are used and most contain fewer than 5 levels. However, the worst-case scenario requires approximately 2x1010 gigabytes to merely store the candidate set. The combinatorics problem explains why stochastic search algorithms such as simulated annealing or genetic algorithms are frequently used instead of an exhaustive search against a complete candidate set.

**Model Definition:**

Our goal is to maximize the weighted D-optimality of the design matrix, penalized for missing distributions and impossible variable interactions, and subject to the distributions of each attribute’s levels and interactions, where each attribute’s level is represented by a binary variable.

*Objective Function* (Wanida Limmun, 2012):

where N is the number of observations, δ are vectors of relaxation variables, and X is the design matrix:

With decision variables *A, B, G* representing attributes:

Age: ,

i.e. the age group classification for each patient *i*.

Gender: ,

i.e. the gender classification for each patient *i*.

BMI: ,

i.e. the BMI classification for each patient *i*.

For easier constraint formulation, we can use the Dantzig-Wolfe reformulation to rewrite our integer variables where the capital letter represents the binary variable series replacing an integer variable, and the lowercase letter represents the integer set of levels permissible for the given attribute:

*Subject to*:

Age group proportions:

Gender proportions:

BMI group proportions:

Binary constraints:

Interaction slacks: While not specified in the toy problem, it is entirely possible that we may have interactions specified in the design (i.e., men cannot be pregnant). In these cases, the slacks are the count of the impossible interactions. We will penalize these interaction slacks twice because they are more costly to the design than a missed distribution.

**Theory:**

For our discrete-choice design, the information matrix of an *n*-point design is

, where X is an *n* x *p* design matrix.

Using as variance estimator, where xi represents a row. See (Labadi, 2015; Triefenback, 2008) for more details regarding optimality theory.

To perform a sequential switch, a ‘delta function’ is defined that allows a less expensive update to the objective function value through the determinant of the information matrix as well as a variance estimator for the swap (Triefenback, 2008):

In order to update our objective function at each iteration, we use the value 1+Δ as the ratio between the new and old objective function value. This allows us to pick out row swaps at each iteration that maximize the increase in the objective function. However, we must alter this ratio if we want to penalize the slacks on our proportions in our objective function, while picking out rows that both maximize the increase in the objective function minimize this penalty:

Therefore, we can define a new update criterion:

This criterion allows us to figure out the row swap that maximizes our objective function, given that the slacks are penalized. It also allows us to terminate the algorithm as the improvement *Δp* converges to zero, i.e. the marginal improvement of another swap becomes trivial.

**Modified Fedorov Algorithm:**

We have implemented a modified Fedorov Algorithm (Labadi, 2015; Triefenback, 2008) that considers the slack of distribution constraints (step 3) when performing the iterative state search:

1. Calculate the *candidate set*, the set of all theoretically possible combinations. Because of the possibility of explosive growth with combinatorics, this will not always be feasible.
2. Generate an initial *n*-point design (an arbitrary design with a nonsingular information matrix) that generally obeys distribution constraints
3. Compute M, MT , and the determinate of M
4. Perform an exhaustive search across the design matrix X and the entire candidate set, using the delta function and (xi, xj) to identify the pair of points that maximally improve *D-optimality*, penalizing the slack from the distribution constraints. Perform the swap.  
   If efficiency metric is sufficiently close to optimal (or improvement from variance estimator is sufficiently small), stop. If the iteration limit is reached, stop.
5. Set and return to step 3

**~~Parallelized Modified Fedorov Algorithm:~~**

~~We attempted to then parallelize the modified Fedorov Algorithm as it has ‘embarrassingly parallel’ tasks in the exhaustive search. In step 4 above, it should be possible to calculate (x~~~~i~~~~, x~~~~j~~~~) in parallel. Our R implementation fails, seemingly due to a bug in the doParallel or foreach libraries that prevent passing a reference class object to the parallel environment. While the parallel infrastructure may require too much overhead to outperform the non-parallel version for the toy problem (especially given that we must store (x~~~~i~~~~, x~~~~j~~~~) for each pair and sort the final list), we believe that as the problem size grows, the effects of parallelizing would show significant runtime improvements.~~

**Genetic Algorithm**

Given the infeasibilities associated with running a discrete, exhaustive search on large candidate sets, we have also implemented a genetic algorithm to perform a stochastic search (Wanida Limmun, 2012):

1. Generate the initial herd of size *population* – a list of initial *n*-point designs (arbitrary designs with a nonsingular information matrix) that generally obey distribution constraints
2. Calculate the *D-optimality* of each design in population. Preserve some number of best designs as elites.
3. Breed random pairs of the non-elite stock (i.e., generate 2 new designs with rows selected randomly from each parent):
   1. Select design pairs at random from the herd (Parent A, B)
   2. Generate mask array of random numbers in [0, 1] of size *n* for each child (Child C, D) to identify from which parent to source rows
   3. If mask value is < 0.5, take row from Parent A, else take from Parent B
4. Randomly mutate cells within the children.
   1. For each child, generate mask matrix of random numbers in [0, 1] the same shape as the design matrix
   2. If the mask value of a given cell is below the mutation threshold *alpha* (a hyperparameter), generate a new value in the range of permitted levels. Higher levels of *alpha* increase the mutation rate and favor exploration of the search space.
5. Collect all parents and children in the herd and assess the fitness (*D-optimality*) of each. Sort. Cull the poor performers, maintaining stable *population* size.
6. Otherwise, identify the most fit design of the new generation and compare to best from prior generation. If fitness increase is sufficiently large, increment the number of generations and go back to step 2.   
   Otherwise, stop due to marginal improvement.

~~Steps 2, 3, 4, and 5 are all ‘embarrassingly parallel’, so it is possible to implement a parallelized genetic algorithm to allow faster traversal of the search space or larger populations and more generations.~~ As the genetic algorithm variant does not use the modified Fedorov update function, but instead checks each experimental design for D-optimality, we can also use the genetic algorithm as a litmus test to confirm that the modifications to the update function work as intended.

**Results**

We ran 500 trials of the toy problem for both the modified Fedorov algorithm and the genetic algorithm with lambda = 0 (constraints unpenalized) and another 500 trials with lambda = 1 (penalty applied to penalize distribution and interaction noncompliance). For each trial *n* of 500, we ensured that the initial design was equivalent between the modified Fedorov and genetic algorithms. The results are summarized below (tables 1, 2)

**Table 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lambda = 0 | Average Initial D-optimality | Average iteration count | Average runtime | Average Final D-optimality | Variance Final D-optimality |
| **Modified Fedorov** | 73.52 | 8.13 | 7.65 | 131.04 | 0.00 |
| **Genetic** | 73.52 | 250.00 | 154.97 | 131.04 | 0.00 |

**Table 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lambda = 1 | Average Initial D-optimality | Average iteration count | Average runtime | Average Final D-optimality | Variance Final D-optimality |
| **Modified Fedorov** | 50.32 | 9.63 | 9.04 | 107.22 | 0.00 |
| **Genetic** | 50.32 | 250.00 | 153.54 | 107.22 | 0.00 |



















Importantly, we observe that our modifications to the Fedorov algorithm work in practice – in all cases, the final D-optimality demonstrates improvement. For both lambda=0 and lambda=1, the modified Fedorov algorithm and the genetic algorithm consistently arrive at final designs with the exact same D-optimality, as demonstrated by the variance of 0.00 across all 500 trials.

Considering the differences between lambda=0 and lambda=1, we see that penalization did in fact take place, as demonstrated by the lower initial D-optimality where lambda=1. As there are violations to level balance per the distribution constraints, the final D-optimality is also lower than the case when lambda=0.

Finally, it is interesting to note that the modified Fedorov algorithm had a vastly lower runtime than the genetic algorithm.

**Full Problem Description**

As mentioned previously, the number of attributes in the full problem is generally limited to no more than 25, each with at most 5 levels due to the complexity of the simulation, limited respondent pool, and limited number of experiments possible per respondent. For a naïve Type 2 Diabetes study, we might define patients with 8 attributes, each containing between 2-4 levels:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | Distribution |
| **Age** | Youth | Adult | Elderly |  | 25/50/25 |
| **Gender** | Male | Female |  |  | 50/50 |
| **Race** | Caucasian | African-American | Hispanic | Asian | 40/20/20/20 |
| **BMI** | Underweight | Normal | Obese |  | 25/40/40 |
| **T2D Dx** | No | Yes |  |  | 50/50 |
| **LDL** | Low | Moderate | High |  | 25/50/25 |
| **BP** | Normal | Moderate | High |  | 25/50/25 |
| **A1C** | Normal | Moderate | High | Very High | 25/25/25/25 |

Among the attributes are a Type 2 Diabetes diagnosis (yes or no) and a given patient’s A1C levels (normal, moderate, high, or very high). As T2D diagnosis and A1C are linked, we have the following interaction constraints: (1) Patients with normal A1C *cannot* have T2D, and (2) Patients with very high A1C *must* have T2D.

The candidate set for this problem has 41,472 different combinations. As before, we seek to reduce time-in-survey by creating a fractional-factorial design, this time consisting of 16 unique patient profiles. The 16-profile fractional-factorial design must be as efficient as possible.

**Model Definition:**

where N is the number of observations, δ are vectors of slack variables, and X is the design matrix consisting of *Aj* attributes:

With decision variables *A1…16* representing attributes:

|  |  |  |
| --- | --- | --- |
| Age |  |  |
| Gender |  |  |
| Race |  |  |
| BMI |  |  |
| Diabetes |  |  |
| Stroke |  |  |
| Heart |  |  |
| LDL |  |  |
| BP |  |  |
| A1C |  |  |
| Renal |  |  |
| Creatinine |  |  |
| UACR |  |  |
| Treatment |  |  |
| Heart History |  |  |
| Smoker |  |  |

Again using the Dantzig-Wolfe reformulation to rewrite our integer variables, *Z* represents the binary variable series replacing an integer variable, and *k* represents the integer set of levels permissible for the given attribute:

*Subject to*:

Age group proportions:

Gender proportions:

Race group proportions:

BMI group proportions:

Diabetes proportions:

Stroke proportions:

Heart group proportions:

LDL group proportions:

BP group proportions:

A1C group proportions:

Renal group proportions:

Creatine group proportions:

UACR group proportions:

Treatment group proportions:

Heart History group proportions:

Smoker group proportions:

Interaction Constraints:

**Results**

We ran 25 trials of the larger problem for both the modified Fedorov algorithm and the genetic algorithm with lambda = 0 (constraints unpenalized) and another 25 trials with lambda = 1 (penalty applied to penalize distribution and interaction noncompliance). For each trial *n* of 25, we ensured that the initial design was equivalent between the modified Fedorov and genetic algorithms. The results are summarized below (tables 3, 4)

**Table 3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lambda = 0 | Average Initial D-optimality | Average iteration count | Average runtime | Average Final D-optimality | Variance Final D-optimality |
| **Modified Fedorov** | 61.35 | 19.32 | 36487.38 | 124.43 | 0.07 |
| **Genetic** | 61.35 | 500.00 | 1582.63 | 96.71 | 12.21 |

**Table 4**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lambda = 1 | Average Initial D-optimality | Average iteration count | Average runtime | Average Final D-optimality | Variance Final D-optimality |
| **Modified Fedorov** | -73.97 | 20.08 | 36150.01 | -21.00 | 1.39 |
| **Genetic** | -73.97 | 500.00 | 1553.30 | -38.51 | 10.13 |



















As with the toy problem, we observe that our modifications to the Fedorov algorithm work in practice – in all cases, the final D-optimality demonstrates improvement. However, unlike with the toy problem, the modified Fedorov algorithm outperforms the genetic algorithm, building more optimal final designs, more consistently.

As before, we again see that penalization did in fact take place, as demonstrated by the lower initial D-optimality where lambda=1. In this larger problem, we added interaction constraints in addition to the distribution constraints, and we see that they strongly effect the initial D-optimality. The negative initial D-optimality does not pose a problem for either algorithm, as they simply look for improvements in the positive direction.

**Future Work**

The primary focus of future work lies in runtime optimization. A single iteration of the larger problem with the modified Fedorov algorithm took around half an hour, for a total runtime of approximately 10 hours on average. As problems get larger, this runtime would grow exponentially. Both the modified Fedorov algorithm and the genetic algorithm are easily parallelizable, and parallelizing would take advantage of the multithreaded and multicore processors in modern computers.

The genetic algorithm has a number of hyperparameters (population size, mutation rate, generation limit, convergence criteria) that necessitate further testing and hyperparameter optimization in order to compete with the modified Fedorov algorithm for consistent improvement in final D-optimality.

Finally, it is likely that tweaks to the linear algebra used in the update functions could improve runtimes. Altering the BLAS/LAPACK implementations used to calculate the functions could further improve the computational efficiency.

**Conclusion**

The Fedorov algorithm seeks to optimize the *D-efficiency* of a set of candidates with respect to the set of all possible candidates. There has been no implementation of the Fedorov algorithm which solves the problem of constraining the design matrix to this point. Here, we have outlined an analytical and a numerical method for dealing with constraints in the design of this algorithm. We have found that the proposed analytical method runs into runtime problems with larger candidate sets, but does indeed converge successfully for smaller problems. As well, this implementation of a genetic algorithm demonstrates improved runtimes on larger problems at the expense of guaranteeing maximal D-optimality. There is room for future improvement of the computational efficiency of the algorithm through parallelization, linear algebra and BLAS/LAPACK optimization, and hyperparameter tuning.

# Works Cited

Hauser, J., & Rao, V. (2002, September). Conjoint Analysis, Related Modeling, and Applications. In *IN MARKET RESEARCH AND MODELING: PROGRESS AND PROSPECTS: A TRIBUTE.* Kluwer Academic Publishers.

Kuhfeld, W., Huber, J., & Zwerina, K. (1996, September). *A General Method for Constructing Efficient Choice Designs.* Retrieved October 2018, from https://faculty.fuqua.duke.edu/~jch8/bio/Papers/Zwerina%20Kuhfeld%20Huber.pdf

Labadi, L. A. (2015, February). Some Refinements on Fedorov’s Algorithms for Constructing D-optimal Designs. *Brazilian Journal of Probability and Statistics, 29*, 53-70.

Triefenback, F. (2008). *Design of Experiments: The D-Optimal Approach and Its Implementation As a Computer Algorithm.* Umeå University, Department of Computing Science.

Warren F. Kuhfeld. (2001, January). *Multinomial Logit, Discrete Choice Modeling.* Retrieved October 2018, from https://www.stat.auckland.ac.nz/~balemi/Choice.pdf