

EVOLUTIONARY ALGORITHMS FOR OPTIMAL SAMPLE DESIGN

Charles D. Day

Internal Revenue Service, Statistics of Income Division

KEY WORDS: Genetic Algorithm, Stratified Sampling, Evolutionary Algorithm, Convex Optimization

INTRODUCTION

Traditionally, the problems of finding stratum boundaries and optimal allocations to those strata have been solved serially, with stratum boundaries determined first. Given that the boundaries affect the allocation, it is natural to ask if finding the optimal boundaries and optimal allocation simultaneously, using the same multivariate information, might not produce a better result. Ballin and Barcaroli suggest a combination of tree search and genetic algorithms for simultaneous optimal allocation and stratification on several variables, but their method is restricted to categorical stratification variables. This paper suggests a different approach; one that uses a single continuous stratification variable.

Biological evolution can be viewed as a process of optimizing a species to (or increasing its fitness for) its environment. Evolutionary Algorithms (EAs), sometimes called Genetic Algorithms after their most common variant, adopt biological evolution as a model for computation. These algorithms are applied most often for finding approximate solutions to computationally intractable optimization problems. The work reported on in the author's 2006 paper [17] focused on the design of an EA for solving the multivariate optimal allocation problem and an investigation of the performance of that algorithm on a simple, well-known example.

This paper extends that work, examining the use of a special type of evolutionary algorithm, a cooperative coevolutionary algorithm (CCEA) to simultaneously determine optimal stratum boundaries and a multivariate optimal allocation of sampling units to those strata.

THE STRATUM BOUNDARY AND MULTIVARIATE OPTIMAL ALLOCATION PROBLEMS

One goal of stratified sampling is to increase the precision (reduce the variance) of estimates of population parameters inferred from a sample. All other things being equal, increased homogeneity of the population being sampled works to increase precision. By dividing the population of interest into non-overlapping subpopulations (sampling strata) that are more nearly homogeneous, selecting independent samples from each stratum, and combining estimates from the strata, the statistician can make a more precise estimate than by directly sampling from the population as a whole.

A number of methods have been proposed for approximating optimal strata with the objective of improving the precision of estimates. Examples of such methods include Winkler [1], Dalenius and Hodges [2], Singh [3], Lavallée and Hidioglou [4], Sweet and Sigman [5], and Gunning and Hogan (for skewed populations) [6]. Additional methods have been proposed that used multivariate information [Jewett and Judkins, Pia].

Once stratum boundaries have been defined, the problem arises of how many sample units to allocate to each stratum. If the survey practitioner wishes only to make as precise as possible an estimate for one variable given a fixed cost, or find the minimum cost design to achieve a target variance, this problem has a well-known solution [7]:

$$n_h = n \frac{N_h S_h / \sqrt{c_h}}{\sum (N_h S_h / \sqrt{c_h})}$$

where n_h is the number of sample units allocated to stratum h , N_h is the number of population units in stratum h , c_h is the cost per unit in stratum h , S_h is the population standard deviation for the variable of interest in stratum h , and n is the total sample size. (S_h is usually estimated from frame information or earlier samples.) If a target variance is fixed and cost is to be minimized, then:

$$n = \frac{(\sum W_h S_h \sqrt{c_h}) \sum W_h S_h / \sqrt{c_h}}{V + (1/N) \sum W_h S_h^2}$$

where $W = N_h/N$. If cost is fixed and variance is to be minimized then:

$$n = \frac{(C - c_0) \sum N_h S_h / \sqrt{c_h}}{\sum N_h S_h \sqrt{c_h}}$$

While it is rarely the case that a survey is conducted to estimate the value of only one variable, this formula is still broadly useful, since an allocation that is optimal for one variable may be near-optimal for variables that are strongly correlated with it. If, however, precise estimates of several variables are needed, and those variables are not all highly correlated with each other, it is desirable to have a method to find a good compromise allocation that will give adequate precision for all of the variables of interest. This is the usual goal of multivariate optimal allocation. A number of approaches have been used to find the optimal allocation on multiple variables [8-12].

There are two common ways to approach this problem. One is to minimize a weighted sum of the variances of the variables of interest. Khan and Ahsan [13] propose a method in which they formulate this problem as a nonlinear programming problem and use a dynamic programming technique to find a solution. One problem with this approach is how to weight the variances. There is no single solution for doing this, and it is not always easy to predict what the consequences of a particular choice of weights are.

The other approach is to choose an acceptable coefficient of variation for each of the variables on which the allocation is to be done. These become constraints on a cost function that can be minimized, giving the following convex programming problem:

$$\begin{aligned} \text{Min:} & \quad \sum c_h n_h \\ \text{s.t.} & \quad \sum_{h=1}^H W_h^2 S_{hj}^2 / n_h t_j^2 \bar{Y}_j^2 \leq 1 \quad \text{for every } j \\ & \quad n > 0 \end{aligned}$$

Where t_j is the target coefficient of variation (CV) of the j th variable and \bar{Y}_j is the population mean of j th variable [14]. It is this second approach that will be adopted in this paper; thus, the problem of interest may be stated as, "Find a combination of stratum boundaries and allocations to those strata that minimizes the budget necessary to achieve predetermined maximum allowable coefficients of variation for two or more selected variables of interest."

EVOLUTIONARY ALGORITHMS

Briefly, evolutionary algorithms adopt biological evolution as a model for computing. While there are a number of canonical variants of evolutionary algorithms, it is common for practitioners to adapt features of two or more variants to develop algorithms specific to the solution of their problems.

In general, evolutionary algorithms start with a "population." Each individual in the population consists of one candidate solution for the problem the EA is trying to solve. Borrowing terminology from biology, each variable in a solution is referred to as a gene, the value for each gene is called an allele, and the structure of the whole solution is referred to as a genome. These candidate solutions are usually generated at random from the space (or a well-chosen subspace) of all possible solutions.

The "fitness" of each individual is then evaluated; that is, the value of the objective function of the optimization problem being solved is determined for each candidate solution. Next, pairs (or n-tuples, should the practitioner wish) of individuals are selected to "reproduce" (reproductive selection). This selection is done in such a way as to favor fitter individuals; for example, individuals could be selected with probability proportional to their fitnesses. Note that the degree to which selection favors fitter individuals controls, in part, the rate at which the algorithm converges. If, say, a few of the fittest

individuals are given a great deal (or all) of the probability of being selected, then only the areas of the solution space in which these lie will be explored. If they lie near local (but not a global) optima, then it is possible to converge rapidly to a less than optimal solution. On the other hand, if there is near-uniform probability of selection with respect to fitness, then there is little pressure to converge toward higher fitness solutions, and the algorithm will fail to converge to an optimum. This balance between “exploration” and “exploitation” is an important design criterion for an evolutionary algorithm.

During reproduction, two operations can be used to produce “children” (the next “generation” of candidate solutions). One consists of taking one part of one of the individuals selected to reproduce and appending it to the complementary part of the individual it was paired with during selection. This is referred to as “crossover” in the EA literature, and is analogous to recombination in biological reproduction (Figure 1).. The second reproductive operator is mutation. As one might

One-point Crossover

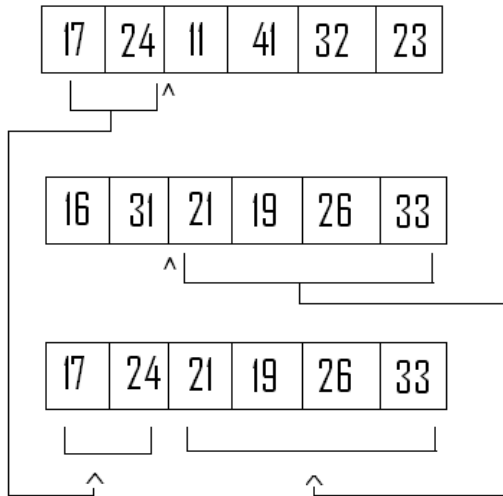


Figure 1. An example of one-point crossover.

suspect, it consists in changing the value of one or more genes in a single individual with some probability. Following reproduction, each child’s fitness is assessed. Children are allowed to survive into the next generation (where they become the initial population) based on their fitnesses (survival selection).

This process continues, with the children becoming the next generation’s parents, until some convergence criterion is reached, or a maximum number of generations is reached. One problem with EAs as described to this point is that the best solution may be lost; that is, the solution with the overall highest (if maximizing) or lowest (if minimizing) value of the objective function may disappear as the algorithm moves from generation to generation, never to be seen again. To address this problem, practitioners usually employ “elitism,” allowing the k highest valued members of the current parent population to survive into the next generation.

Should the reader wish a thorough introduction to the field of Evolutionary Algorithms, De Jong [14] provides one.

Cooperative Coevolution

In nature, a species rarely evolves without being affected by a relationship with one or more other species. That relationship may be competitive; for example, a predator and its prey, or two species that compete for the same food source. In other cases, the relationship may be symbiotic, such as flowers and the insects that pollinate them. Over time, both the flowers and the insects have evolved together, or coevolved, to improve the effectiveness of their relationship.

Wiegand [15] defines a coevolutionary algorithm as “an evolutionary algorithm that employs a subjective internal measure for fitness.” By subjective, Wiegand means that the fitness (value of the objective function) of an individual cannot be evaluated independently, but rather must be computed using one or more other individuals (collaborators). For example, as

we will define fitness, it will not be possible to evaluate the fitness of an allocation without doing so in reference to a particular set of stratum boundaries.

By internal, Wiegand means that the fitness influences the course of evolution in some way. The value of a candidate solution on an objective function is an example of an internal fitness when it is used to make that individual more or less likely to survive or reproduce. Complex optimization problems may be modeled by decomposing them into parts and representing candidate solutions to each of the parts as coevolving species in an evolutionary algorithm. Designing a successful CCEA is not a trivial exercise. The following discussion of the relevant issues draws heavily from Wiegand. First, one must decide how to represent a candidate solution to his or her problem. Choice of representation is often critical to the success of an EA.

Fortunately, for CCEAs, there are some principles to guide the designer. In particular, it is important that the separation of the candidate representation into coevolving parts matches the natural decomposition of the problem. If the designer creates more coevolving populations than there are parts to the problem, this results in strong interactions between those populations representing the same part. This condition is termed “cross-population epistasis,” a characteristic that often causes poor performance of the algorithm.

Once a representation is decided upon, it is necessary to decide how the coevolving populations will interact. Since CCEAs have subjective fitness, when fitness is evaluated it must be done on the space of interactions between the populations. It may be tempting to try and evaluate each member of one population in collaboration with each member of the coevolving population, so-called “complete mixing.” Note that, if the populations are relatively large, this requires an extremely large number of fitness evaluations, usually the most computationally expensive part of an EA. In most circumstances, evaluation of a sample from the interaction space for each individual is sufficient.

Note that, given the space of interactions between all possible collaborators from both populations, the current populations restrict fitness evaluation for any individual in one population to the “projection” of the individuals in the coevolving population onto the interaction space. That is, in a two-population CCEA, population A is optimized for collaboration with population B, and population B is optimized for collaboration with population A. As the two populations evolve, they may eventually reach what is referred to as “robust resting balance.” Readers familiar with game theory may recognize this as an example of a Nash equilibrium. Nash equilibria, like the optimal strategy chosen by each individual in the familiar Prisoner’s Dilemma game, are not guaranteed to be the global optimum solution. Wiegand suggests some design choices that are incorporated in the design of the algorithm for this investigation to ameliorate this problem.

DESIGN OF A CCEA TO SOLVE THE MULTIVARIATE OPTIMAL STRATUM BOUNDARY CONSTRUCTION AND SAMPLE ALLOCATION PROBLEM

Given the interaction between stratum boundaries and optimal allocations, one may view the construction of boundaries and allocation of sample units as a single, decomposable problem, with the obvious natural decomposition. For purposes of this investigation, two populations, one of stratum boundaries and the other of allocations to those strata are allowed to coevolve.

Optimal Allocation

When designing an EA (or any other optimization algorithm), it is important to incorporate any special features of the problem to be solved. In the case of optimal allocation, any solution that results in enough of the available budget (total cost) being left over to allocate another unit in any stratum is sub-optimal; that is, any optimal solution must use the entire budget. This implies that, rather than searching the entire space of feasible allocations, the EA can concentrate on a bounding hyperplane of that space, enormously reducing the size of the set to be searched and reducing time to convergence. This can be incorporated into the problem as a constraint. A second constraint consists of the need to meet the maximum CV targets. The algorithm described here takes a conventional approach to the first constraint and an unusual one to the second.

By defining the allocations to the strata as budget allocations (making the number of units allocated implied rather than explicit), and incorporating the budget constraint into the initialization, mutation, and crossover operators by calling a repair function at the end of each operation, the algorithm constrains the search to the hyperplane in which the solution must lie. When creating an initial population of candidate allocations, the initialization operator chooses gene values at random from a fixed range with uniform probability. The operator then calls a function which finds the sum of the allocated values and normalizes the vector of allocations so that the sum of its elements equals the budget with the constraint that no stratum may be assigned an allocation less than two. So, the EA starts with an initial population of vectors that lie in the hyperplane with

the optimal solution. If the search is to be constrained to that hyperplane, mutation and crossover operators must operate to keep children in that region of the solution space as well. The mutation operator uses a parameter that contains the probability of mutation to decide whether or not to mutate a particular gene. If a gene is chosen for mutation, its value is increased or decreased (with equal probability) by the cost of one unit in that stratum. The repair function is then called again to ensure that the budget is maintained. The crossover operator is relatively simple, doing a simple one-point crossover with tournament selection and calling the repair function should the budget constraint be violated.

The phrase “tournament selection” needs a little explanation. As discussed earlier, in any EA it is necessary to select individuals reproduce. These selections are done in such a way that fitter individuals have a greater chance of being chosen to reproduce, thus putting pressure on the whole system to evolve fitter and fitter individuals in successive generations. One method of choosing individuals to reproduce is to hold a “tournament” in which k individuals are chosen at random and the fittest is selected to reproduce. The larger k gets, the more likely the randomly chosen participants are to contain at least one high fitness individual; therefore, larger values of k are associated with greater selection pressure. In this application, exploration of the solution space was heavily valued over rapid exploitation of promising areas, thus k was set at its minimum value of two.

Optimal Stratification

What concerns us in this section is not development of a general method for determining multivariate optimal stratum boundaries. We are instead concerned with the development of a complementary algorithm to the optimal allocation algorithm described above; that is, one that determines optimal stratum boundaries given an allocation.

Fortunately, we were able to borrow considerably from the optimal allocation algorithm. For example, the fitness function is essentially the same, except that instead of there being a different vector of n_{hs} for each individual, the individuals undergoing evolution are integer vectors of AGI boundaries, similar to the boundaries developed by the Lavallée and Hidiroglou method used for comparison below, but represented a little differently. Given the n_{hs} as fixed, the boundaries are used to compute the stratum standard deviations for each study variable, and the W_{hs} for each stratum. These are the quantities in the fitness function that vary for each candidate vector of boundaries.

Some thought was needed about how to implement the fitness function. Consider an EA that uses a breeding population of size 100 and evolves it over 1,000 generations (a tiny number by EA standards). This implies that ~100,000 fitness evaluations will be done. If the evaluation process is slow, then the evolution will be glacial. A simple-minded approach, in which the data file must be read once for each evaluation in order to estimate the stratum standard deviations is prohibitively expensive in terms of computing time.

Consider the usual computational formula for S:

$$\sqrt{\frac{\sum x_i^2 - \frac{(\sum x_i)^2}{N}}{N}}$$

If the data are sorted on ascending value of the stratifier, then cumulative $\sum x_i$ and $\sum x_i^2$ can be pre-computed and stored in a file. These can be read in once per generation, when the fitness evaluation object is instantiated. By representing the stratum boundaries not as values of AGI, but as row indexes into the data matrix sorted on AGI, it is a simple matter to calculate the stratum $\sum x_i$ and $\sum x_i^2$ by subtraction of respective cumulative total at the lower boundary of the stratum from that at the higher boundary. This greatly speeds and simplifies the calculation of fitnesses. A more sophisticated method will have to be found if one wishes to use multiple continuous stratifiers.

Initialization was done by generating an integer vector of five stratum boundaries represented as row indexes into the data matrix sorted on the stratifier, AGI. These boundaries determined the strata sampled at less than 100 percent. The final value of the allocation vector was used to determine the lower the boundary of the take-all stratum. Candidate solutions in which that boundary was below the fifth and last non-100 percent boundary were eliminated using a very large penalty in the fitness function. The condition that needed to be maintained in this case was that stratum boundaries must be in ascending order,

otherwise population units are included in two or more strata. So, after boundary vectors were instantiated, their values were sorted to preserve the ascending order property. This is, in effect, another simple repair function, and was used after mutation and crossover as well. Mutation was initially done similarly to the method for optimal allocations, except 100 was added or subtracted instead of 1, as the numbers being dealt with were approximately two orders of magnitude larger. This method proved unsatisfactory, as 100 proved too small in the initial generations when the algorithm needs to search the solution space more broadly, and too large in later generations when the algorithm was trying to converge. An idea was borrowed from simulated annealing. As the number of generations became larger, the mutation increment was reduced, starting with 2000 and ending with ten. One-point crossover was also employed, with Tournament selection.

Selection for reproduction was done using the roulette-wheel method. This method selects an individual i from the population with probability proportional to its fitness, similar to a PPS sample of one with fitness as the size measure. As in most implementations, elitism is employed to avoid losing the best solution found to that point.

The fitness function is modeled after the objective function used by Bethel [16]. Given a vector of allocations to strata, the program calculates a “standardized precision unit” (SPU) for each variable j as follows:

$$SPU = \sum_{i=1}^H W_i^2 S_{ij}^2 / n_i t_j^2 \bar{Y}_j^2$$

Note that this is the left-hand-side of the first constraint on the cost minimization problem discussed earlier. Further, the variance constraint on the j th variable is met when this quantity is equal to one. Using the SPUs a fitness function can be formed. This EA uses:

$$fitness = \sum_{SPU_j > 1} SPU_j^2$$

as its fitness function. This function is minimized.

The EA described here is actually a *framework* for an optimization algorithm. The CV constraint is treated as the objective function, with the budget a fixed parameter for any one run of the algorithm. This is not the objective for the optimal allocation problem as defined here. That objective is to *minimize the sample size*. To achieve that objective, the EA is used in a binary search over the set of sample sizes, with each set of EA runs used as a test to determine whether or not a solution meeting the target CVs can be found given the sample size being tested. If a solution is found, then a size from the lower interval defined in the binary search is explored. If, after a sufficient number of unsuccessful runs of the EA have been completed to convince the user that no solution can be found with the current sample size, then the upper interval defined in the binary search is explored. This continues in the usual fashion for a binary search algorithm until the binary search converges.

DESCRIPTION OF THE TEST DATA

Testing the CCEA method’s performance required an example problem, results on which could be compared to more standard methods. The author’s primary interest is in economic data, particularly tax data, and the example problem is drawn from this field. Confidentiality constraints prevent the use of live data, so a set of synthetic data were constructed. Four aggregate income variables were created by treating the Statistics of Income Individual Tax Return sample file as a population. Note that the SOI sample is stratified, so the unweighted distributions of amount variables in the sample does not represent the true distributions of these variables. (They are considerably more skewed.) Thus, despite the variable names, the characteristics of these data should not be taken to represent real world statistics.

For each of the four variables, Wage Income, Business Income, Investment Income, and Taxable Retirement Income, a univariate lognormal distribution was fit to the unweighted SOI sample data. In addition, a residual amount was computed by subtracting the components of the four named variables from Adjusted Gross Income (AGI), and a normal distribution was fit to these residuals to represent other components of AGI.

In order to create an observation, a random variate was generated from the fitted distribution for each of the four variables of interest which was to be present in that observation. In addition, a residual amount was generated from the residual distribution. Not all of the four named variables are present on every tax return; therefore, to add a real world complication

to the example, patterns of presence or absence from the real data were used to create a vector of indicator variables for each synthetic observation. AGI was computed by summing the generated values of the four variables of interest times their respective indicators and the residual variate. Note that no attempt was made to preserve multivariate relationships that might exist between these variables on real tax returns. This provides an additional level of confidentiality protection. The distributions of the synthetic data amounts are presented in Figures 2-5.

Despite care having been taken to prevent these variables from too closely mimicking real tax data, the data set has several interesting properties for evaluation of sample stratification and allocation methods. Figure 6 shows the proportion of records with each of the four variables of interest and their correlations with the stratifier, AGI. Three of the four possible combinations of high and low frequency of presence and high and low correlation with AGI are present. Only the easiest situation, a variable present in a high proportion of observations that is highly correlated with AGI is not represented. Given

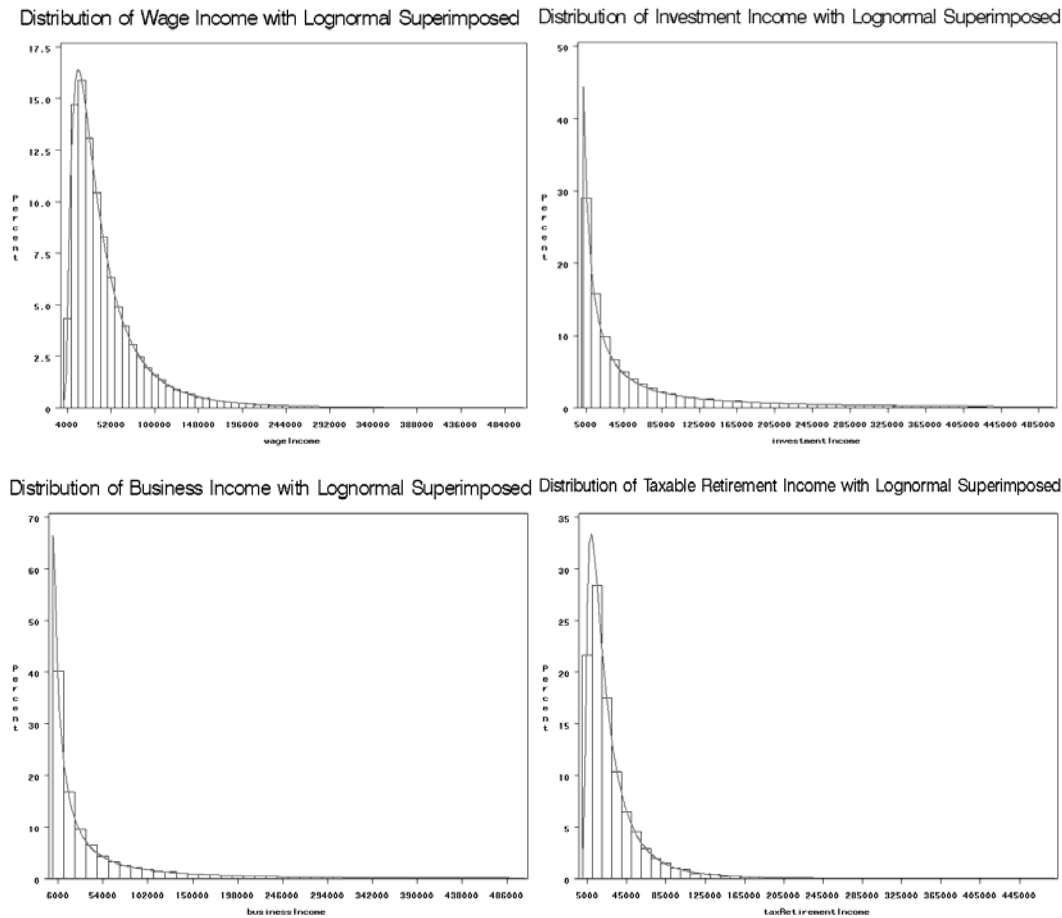


Figure 2. Distribution of Synthetic Variables

these characteristics, along with the highly skewed nature of the variables of interest and the stratifier, the example problem data represent a fair test of a method's ability to perform in real world conditions.

RESULTS

The results of any new method should be compared with methods already in common use. The method described by Lavallée and Hidiroglou (L-H) [4] is one such method for stratification and allocation in the presence of skewed distributions. Table 2 shows the stratification on AGI and Neyman allocation to achieve a CV of 0.10 resulting from the use of the L-H method. Table 3 shows the sample sizes and CVs using the L-H strata for Neyman allocation, multivariate optimal

allocation with target CVs of 0.10 for all four variables of interest using the L-H sample size, and the same thing using the minimum sample size necessary to achieve the target CVs.

Table 4 shows the stratum boundaries and allocations found using the method described in this paper (CCEA), while Table 5 shows the associated CVs. The first thing to note is that the CCEA method can match the performance of serial stratification using the L-H method and multivariate optimal allocation. Additional test runs with smaller sample sizes were disappointing. While CVs very close to the targets could be obtained, no runs were observed in which the target CV for Taxable Retirement Income was met. For these four variables, no improvement over commonly used methods was obtained.

Variable Name	Correlation with AGI	Percent Observations Present
Wage Income	0.0844	83
Business Income	0.4181	27
Investment Income	0.9021	31
Taxable Ret. Income	0.0721	29

Table 1. Correlations with AGI and percentage of records with a nonzero value of the variable for synthetic data.

Why might this be? Note that Taxable Retirement Income shows a correlation of only 0.0721 with the stratifier; that is, the stratified sample can be expected to be only slightly, if any, better than a simple random sample. Thus, it's not unreasonable to believe that the CCEA's ability to alter stratum boundaries will have little effect on the needed sample size to reach the target CV for this variable.

What happens if this variable is eliminated? Table 6 shows the results of multivariate optimal allocation using the L-H stratum boundaries on the three remaining variables. Note the drop in the required sample size to meet the remaining CV targets from 475 to 346. Runs with the CCEA method were more encouraging, with the CCEA method able to find a solution meeting the remaining CV targets with a sample size of only 330 (Tables 7 and 8). The ability to adjust strata and allocations simultaneously led to a smaller minimum sample size to hit the CV targets.

CONCLUSIONS

Use of a cooperative coevolutionary algorithm to find simultaneous optimal stratum boundaries and multivariate optimal allocations is effective in reducing the sample size needed to meet CV targets for variables with moderate or stronger correlations to the stratifier. For multivariate optimal allocations in which at least one variable of interest is poorly correlated with the stratifier, the CCEA method is as good as the commonly used Lavallée and Hidiroglou method for finding stratum boundaries to be used for multivariate optimal allocation.

RECOMMENDATIONS

The results of this investigation are sufficiently encouraging to warrant additional research in the CCEA method. CCEAs should be compared to more stratification methods, in particular the Gunning and Horgan method. Further, CCEAs should be tested against other methods used for sampling from different distributions, including bimodal, multimodal, and other mixture distributions. The lack of any distributional assumptions by the CCEA method makes it use for designing samples from unusual or unknown distributions of interest.

Stratum Upper Boundary (AGI)	Population (N = 119,326)	Sample Allocation (n = 58)
33,000	37,785	3
71,415	36,118	4
152,761	26,960	6
405,364	13,629	10
1,619,751	4,154	13
17,976,308	674	16
59,528,121	6	6

Study Variable	Allocation Method		
	Neyman on AGI (Default L-H, n = 58)	Multivariate Optimal (n = 58)	Multivariate Optimal (n = 475)
Wage Income	0.146	0.116	0.040
Business Income	0.282	0.265	0.100
Investment Income	0.129	0.169	0.066
Taxable Retirement Income	0.678	0.296	0.100

Tables 2 and 3. Results of Lavallée and Hidioglou Stratification and Three Approaches to Optimal Allocation.

Stratum Upper Boundary (AGI)	Population (N = 119,326)	Sample Allocation (n = 475)
31,196	35,338	36
58,402	29,529	56
107,940	26,130	91
277,033	20,395	132
1,392,710	7,359	104
19,753,607	841	52
59,528,121	4	4

Table 4. CCEA Multivariate Optimal Stratum Boundaries and Allocations

Variable Name	Coefficient of Variation
Wage Income	0.040
Business Income	0.100
Investment Income	.0071
Taxable Retirement Income	0.100

Table 5. CVs for CCEA Multivariate Optimal Allocations and Stratum Boundaries, n = 475

Stratum Upper Boundary (AGI)	Population (N = 119,326)	Sample Allocation (n = 346)
33,000	37,785	13
71,415	36,118	28
152,761	26,960	52
405,364	13,629	83
1,619,751	4,154	95
17,976,308	674	69
59,528,121	6	6

Table 6. Multivariate Optimal Allocation Using Lavallée and Hidiroglou strata, dropping Taxable Retirement Income

Stratum Upper Boundary (AGI)	Population (N = 119,326)	Sample Allocation (n = 330)
55,082	61,836	29
124,239	33,332	48
252,241	14,898	58
551,708	6,121	61
1,680,115	2,503	68
19,753,607	632	62
59,528,121	4	4

Table 7. CCEA Multivariate Optimal Allocation and Stratum Boundaries dropping Taxable Retirement Income

Study Variable	Method (Sample Size)	
	Lavallée and Hidirouglou (n = 346)	Cooperative Coevolutionary Algorithm (n = 330)
Wage Income	0.057	0.060
Business Income	0.100	0.100
Investment Income	0.061	0.061

Table 8. CVs for L-H with Multivariate Optimal Allocation and CCEA methods dropping Taxable Retirement Income and meeting remaining 3 CV targets.

ACKNOWLEDGEMENTS

The author thanks the Statistics of Income Division, Internal Revenue Service for supporting this work. In addition, a debt of thanks is owed to Sean Luke and a number of graduate students in the Computer Science Department at George Mason University for their development of the Evolutionary Computation in Java (ECJ) library [19], which was used to develop the programs employed in this work.

REFERENCES

- [1] Winkler, W. E. (2004) Sample Allocation and Stratification. Chapter in *Handbook of Sampling Techniques and Analysis*, unpublished.
- [2] Dalenius, T., and Hodges, J. L. (1959) Minimum Variance Stratification. *Journal of the American Statistical Association*, Vol. 54, pp. 88-101.
- [3] Singh, R. (1971) Approximately Optimum Stratification on the Auxiliary Variable. *Journal of the American Statistical Association*, Vol. 66, pp. 829-833
- [4] Lavallée, P., and Hidirouglou, M. A. (1988) On the stratification of skewed populations. *Survey Methodology*, Vol. 14, pp. 33-43.
- [5] Sweet, E. M. and Sigman, R. S. Evaluation of Model-Assisted Procedures for Stratifying Skewed Populations Using Auxiliary Data. In *Proceedings of the Section on Survey Research Methods, 1995*. American Statistical Association, 1995, pp. 491-496.
- [6] Gunning, P. and Horgan, J. M. A New Algorithm for the Construction of Stratum Boundaries in Skewed Populations. *Survey Methodology*, Vol. 30, pp. 159-166.
- [7] Cochran, W. G. *Sampling Techniques*, 3rd edition, John Wiley and Sons, New York, NY, 1977, pp. 97-98.
- [8] Kokan, A. R. and Khan, S. (1967) Optimum Allocation in Multivariate Surveys: An Analytical Solution. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 29, pp. 115-125
- [9] Huddleston, H. F., Claypool, P. L., and Hocking, R. R. (1970) Optimal Sample Allocation to Strata Using Convex Programming. *Applied Statistics*, Vol. 19, pp. 273-278.

- [10] Kish, L. (1976) Optima and Proxima in Linear Sample Designs. *Journal of the Royal Statistical Society. Series A*, Vol. 159, pp. 80-95
- [11] Chromy, J. B. Design Optimization With Multiple Objectives. In *Proceedings of the Section on Survey Research Methods, 1987*. American Statistical Association, pp. 194-199.
- [12] Bethel, J. (1989) Sample Allocation in Multivariate Surveys. *Survey Methodology*, Vol. 15, pp. 47-57.
- [13] Khan, M. G. M. and Ahsan, M. J. A Note on Optimum Allocation in Multivariate Stratified Sampling. *South Pacific Journal of Natural Science*, Vol. 21, pp. 91-95.
- [14] DeJong, K. A. (2006) *Evolutionary Computation: A Unified Approach*. MIT Press, Boston, MA.
- [15] Wiegand, R. P. (2004) An Analysis of Cooperative Coevolutionary Algorithms. Ph.D. Dissertation, George Mason University, Fairfax, VA. <http://www.tesseract.org/paul/papers/rpw-dissertation.pdf>
- [16] Bethel, J. *op. cit.*
- [17] Day, C. Application of an Evolutionary Algorithm to Multivariate Optimal Allocation in Stratified Sample Designs. *2006 Proceedings of the American Statistical Association*, Survey Research Methods Section [CD-ROM]. Alexandria, VA: American Statistical Association
- [18] Luke, S. et al. Evolutionary Computation in Java (ECJ), <http://www.cs.gmu.edu/~eclab/projects/ecj/>.