STUDY SERIES
(*Statistics #2005-02)*

Protecting Sensitive Cells in a Cell Suppression Program Using Sliding Protection

Paul B. Massell

Statistical Research Division
U.S. Bureau of the Census
Washington D.C. 20233

Report Issued: January 31, 2005

Disclaimer: This report is released to inform interested parties of ongoing research and to encourage discussion of work in progress. The views expressed are those of the author and not necessarily those of the U.S. Census Bureau.

STUDY SERIES (Statistics #2005-02)

Protecting Sensitive Cells in a Cell Suppression Program Using Sliding Protection

Paul B. Massell

Abstract

Cell suppression has been a commonly used method at the Census Bureau and at other agencies for protecting sensitive cells in statistical tables whose cells contain magnitude data. In this method, the sensitivity of each cell depends on the distribution of the respondent values which are summed to form the cell value. Those cells determined to be sensitive are suppressed and then a cell suppression program is run to determine which additional cells (called secondary suppressions) need to be suppressed in order to protect the sensitive ones. In this study, we compare two ways of protecting sensitive cells and their effects on the suppression patterns, i.e., the set of secondary ones. These ways are (1) fixed interval protection and (2) sliding protection. In studies done over a decade ago by researchers at the University of Maryland, it was shown that sliding protection often leads to fewer secondary suppressions than fixed interval protection. Here we show that this result does **not** hold when the cell suppression program incorporates the following assumption: if v is any respondent value, then any table user knows, from publicly available information, that the value lies in the interval [0, 2*v]. In other words, respondent values are known by interested parties to within 100% of their actual value.

Keywords: cell suppression, sliding protection, fixed interval protection

Table of Contents

1. Introduction

We assume the reader has some knowledge of cell suppression, ways of determining sensitive cells (primary suppressions) and the need for finding secondary suppressions (complementary suppressions) that protect the sensitive cells after they have been suppressed. (See references [J] and [M] for background material). Suppose one applies a linear sensitivity measure, such as the p% rule, to determine which cells in a table about to be released by an agency, require disclosure protection. Suppose that for a particular sensitive cell, 'prot' is the amount of protection required. We use the term 'fixed interval protection' for a process which creates an uncertainty interval for any sensitive cell that has at least 'prot' units on each side of the actual value. This could also be called 'two-sided protection'. Sliding protection is an alternative to fixed interval protection. It requires only that the uncertainty interval created for each respondent value be at least 2*prot units wide but does not specify any width for each side of the actual value. It has some advantages and disadvantages with respect to fixed interval protection. In cell suppression programs that implicitly assume that table users know only that respondent values are non-negative, sliding protection requires fewer secondary suppressions but provides what some users might consider a slightly weaker form of protection of respondent values. In this paper, we compare fixed interval protection with sliding protection in cell suppression programs that incorporate a reasonable assumption about knowledge that table users have about respondent values prior to using the tables about to be released. It turns out that such an assumption affects the relationship between sliding and fixed interval protection.

2. Mathematical definitions; policy issues

In fixed interval protection, after one determines that a cell is sensitive and how much protection it requires (often accomplished using the p% rule), one uses a suppression program that finds a complementary cell suppression pattern (i.e., the set of 'C' cells). The pattern that the suppression program finds has the property that it provides the amount of protection required on BOTH sides of the true value. For example, suppose there are only two contributions to a cell, say \$200 and \$80, and p=10. The p% rule says that protection required by the top contribution is (p/100) time its value minus S where S is the sum of all the contributions starting with the 3rd largest to the smallest. Since S=0 in this example, the protection needed is (10/100)*\$200 = \$20. The pattern found using fixed interval protection will ensure that the tightest interval that a table user can derive from the table of the top contributor's value is (180, 220). Often the actual uncertainty interval (sometimes called a **feasibility** interval, derived from the term's usage in mathematical programming) will be somewhat larger or even much larger than this interval. Often it will **not** be symmetric about the true value.

In contrast, if one uses sliding protection on the example above, the pattern created will create a feasibility interval of the form (a, b) where a is less than or equal to 200, b is greater than or equal to 200, and the width of the interval, (b-a), is at least 40. That is, the minimal width (of the feasibility interval) for sliding protection is the same as the minimal width for fixed interval protection; in the example above it is 40. The main difference with the protection methods is that sliding protection often produces an interval is which **one** of the endpoints is less than the protection distance (20, in the fixed interval example) from the true value.

There may well be occasions when the true value equals one of the end points of the protection interval. Given this possibility, a policy decision must be made regarding whether sliding protection provides adequate protection. If it is decided that it does provide adequate protection, then one needs to see if there are some advantages to using it rather than fixed interval protection.

3. Practical implementation issues

Currently our suppression programs have implemented fixed interval protection. Creating a linear programming (LP) based suppression program that can implement sliding protection does not require extensive program changes but the changes require delicate programming. However, the suppression program that is used for much of the production work at the Census Bureau (especially for large tables) is based on the network flow method, which is a very fast special case of linear programming. For this program, implementing a sliding protection capability may be a bit more complicated just because of the structure of the program (ref. [J]). The number of variables required to implement sliding protection is twice the number required for fixed interval protection. Since mathematical programs of a given complexity usually have runtimes that increase at least linearly with the number of variables, the sliding protection programs are likely to take longer than the comparable fixed interval program.

4. Problems explored in this study

Let us give a little historical background. In the early 1990's, the Census Bureau let a contract to professors at the University of Maryland Business School, Bruce L. Golden and Arjang A. Assad and a graduate student, James P. Kelly. They wrote some technical papers in which they did a comparative analysis of fixed interval protection versus sliding protection under some general conditions. In one of their papers [KGA, 1992], they found that sliding protection 'can significantly reduce the total amount of suppressed data'.

After trying unsuccessfully to replicate these earlier results, we noticed that there was a difference between the KGA cell suppression program and the Census production program, regarding how much 'protection flow' is allowed to flow through a given secondary suppression. The upper bound on 'protection flow' is based on the assumption about table user knowledge of respondent values. To test exactly how this knowledge assumption affects sliding protection, we introduced a new parameter, 'capmult' (for 'capacity multiplier') that we could change before each run. It was used to impose an upper bound on the flow variables;

$$\text{ub(i)} \leq \text{capmult} \bullet \text{capacity(i)}$$

where ub(i) is the upper bound of protection flow allowed through the ith cell and capacity(i) is the capacity of the ith cell to protect the sensitive cell being protected (ref. [J]). The simplest case, which often occurs, is one where the capacity of a cell equals its value. Our main observation is that this upper bound limits the amount of sliding that can take place. In particular, the value capmult=1 corresponds to the case in which the table user can estimate any respondent's true value within 100% on the low or high side. Thus if the true value is '40', the user knows only that the true value lies in the interval [0, 80]. A knowledge assumption that is equivalent to setting capmult=1 is built into Census production suppression programs.

If one assumes that a table user will know only that a respondent's true value is non-negative, then one would set capmult equal to a very large value that represents infinity for computational purposes. This is the case implicit in the KGA work. Of course, one should use a value of capmult that seems to approximate the knowledge of respondent values that the most informed table users are likely to have based on various data sources, both governmental and private. Subject matter specialists may be able to suggest a realistic value of capmult. Ideally one could conduct a survey of likely table users to estimate a reasonable value. In addition, it has been suggested that a 2nd parameter is needed to reflect the lower bound of a table user's a priori knowledge of a respondent value more realistically than simply assuming the value is non-negative. However, Census suppression programs do not currently do this and we do not discuss this two parameter case in the paper.

One can show that for this value, no sliding of protection intervals is allowed, therefore fixed interval protection (FP) is equivalent to sliding protection (SP) (i.e., FP = SP) when capmult=1. The argument goes as follows. Using the capmult=1, leads to symmetric bounds on the protection flow for each suppressed cell, i.e., the maximum amount of negative flow allowed equals the maximum amount of positive flow allowed through each cell. This symmetry means that any flow through the set of suppressed cells can be reversed. This reversibility of the flow means that if we have x units of protection in one direction for a given sensitive cell, then we must have x units of protection in the other direction for the same cell. That is, for the simple example given in section 2, if we have 20 units of protection on one side of 200 we will also automatically have 20 units of protection on the other side. Thus the protection created with capmult=1 is guaranteed to have fixed interval (i.e., two-sided) protection.

Our proof of reversibility of protection flow for the case capmult=1, holds for the suppression pattern generated for a single sensitive cell. When one protects a set of sensitive cells sequentially, the feasibility interval may lose its symmetry but the cell still has fixed interval protection with each side of the uncertainty interval having at least 'prot' units of protection. The backtracking procedure, performed on a set of linked tables, can also lead to asymmetric intervals but preserves fixed interval protection.

We decided it was worthwhile to explore the case capmult=100 to see if our results were comparable to the earlier work done at the University of Maryland. To that end, we did the following comparisons:
(i) FP vs SP comparison for 5 separate (i.e., unlinked) tables (i.e. column relations)
(ii) FP vs SP comparison when the column relations are linked; processed on a single run and where, as usual, backtracking is used to provide consistent protection patterns

We also tested the effect of backtracking with linked tables on the FP vs SP comparison.
We tested the effect of SP on the percentage of primary suppressions (called 'P cells') that are well-approximated by the midpoint of the feasibility interval. We computed the percentage of P cells for which one endpoint of the feasibility interval is close to the cell value.

5. Results

In all these tables we set capmult=100 thereby making the sliding protection (essentially) unconstrained. This unconstrained case is the one treated in the KGA references. As discussed in the above section, the Census production suppression programs, in contrast, use, capmult=1 with fixed interval protection.

The quantity we usually want to minimize is the total value of complementary cells (C's). The production cell suppression programs use network flow, which like linear programming, cannot minimize that quantity but is able to minimize a rough approximation to it, viz., the sum over C cells of value(i) * flow(i) where flow(i) is the flow through cell i. This explains why sometimes SP has a larger entry in the 'Total Value of C' column than the FP entry. That is, if one used integer programs in which we could minimize the sum over C cells of value(i), we would expect that the Total Value of C for SP would never be greater than the value for FP.

Comments on Tables 1 - 4.

Tables 1 and 2 have results for the five tables when they are treated as independent tables and processed on five separate runs of the suppression program.

Tables 3 and 4 have results for the five tables when they are treated as linked and suppressed on a single run of the suppression program.

Comparing KGA's results with ours

KGA found a decrease of about 12% in total value of C for a large number of tables, using simulated Census-like data. With our quite limited set of tables we found SP sometimes had a great decrease in Total Value of C's compared to FP; however for some tables FP was lower (see above explanation for this). Thus our result from our small set of tables seems to be consistent with the Kelly et al. finding [KGA, 1990, p.25] that SP allows on average a 12% lower information loss (= value of suppressed cells) based on a sample of over 1000 real-data tables.

Comparing FP with SP regarding vulnerability to the midpoint attack

One might suspect that SP has an advantage compared to FP. Since for SP, two-sided protection is not required, one might guess that the uncertainty interval (i.e., feasibility interval) derived from the complete suppression pattern, would be less likely to be vulnerable to a midpoint attack. By this we mean that the midpoint of the uncertainty interval is less likely to be close to the true value with SP than with FP. To test this we introduced a measure of closeness to the midpoint of the uncertainty interval. We set z = abs(mid-val)/val and say the midpoint (of the uncertainty interval) is 'close to the value' if z is less than some parameter value 'sper'. We set sper=0.05 . With our limited test set, there is no significant difference between SP and FP in the number of P that are well approximated by the midpoint of their feasibility interval as calculated by the audit program.

We also tested the sensitive cells that received sliding protection but did not receive fixed interval protection to see if their value was close to the endpoint of the uncertainty interval. We know that sliding protection allows, in theory, for the value to be close to or even equal to the endpoint of the uncertainty interval. If this case occurs frequently, then sliding protection is creating a great deal of uncertainty for the data intruder. However, we wanted to see how often this case arises with real data tables. We set wu=(upper - value)/width and wl=(value-lower)/width. If either of wu or wl is less than sper2 we say that the value is close to an endpoint of the uncertainty interval. We set sper2=0.05 . For SP, only a small percentage of P's seem to be close to an endpoint of the feasibility interval. (See Results Tables below)

Recall that Tables 3 and 4 have results for the five tables when they are treated as linked and when suppression patterns are found on a single run. A single run in general requires many backtracking passes to ensure that the final suppression patterns in the tables are consistent in cells that are common to two or more tables. It appears that the way backtracking is implemented in our suppression program leads to a very large number of C's for data tables 3 and 4 for SP. In fact there are many more C's for the SP case than for the FP case and the total value of the C's is also much greater for SP case. This contradicts the main reason for using SP; i.e., lower loss of information. This situation needs to be explored more thoroughly to determine if the backtracking procedure is not currently handling SP correctly.

Example 1: Five tables treated as unlinked by processing them on separate runs of a suppression program.

TABLE 1 Number and Value of C's
Fixed Interval vs. Sliding Protection

	Fixed Interval Protection		Sliding Protection	
Data Table	Number of C's	Total Value of C's	Number of C's	Total Value of C's
1	14	84,414	12	26,526
2	284	367,715	270	381,058
3	290	324,742	261	283,916
4	51	175,945	47	143,810
5	219	310,796	187	242,256

TABLE 2 Closeness of P to midpoint of protection interval
Fixed Interval vs. Sliding Protection

	Fixed Interval Protection		Sliding Protection			
Data Table	Number of P's	% of P's close to midpoint	Number of P's	% of P's close to midpoint	Number of P's not receiving fixed prot.	% of P's (prev. col.) close to endpoint
1	4	0%	4	0%	2	100
2	42	19%	42	26%	10	20
3	44	23%	44	23%	8	13
4	9	22%	9	22%	2	100
5	33	30%	33	21%	5	40

Example 2: A set of five linked tables ; processed on a single run; extensive backtracking

TABLE 3 Number and Value of C's
Fixed Interval vs. Sliding Protection

	Fixed Interval Protection			Sliding Protection		
Data Table	# of passes	Number of C's on final pass	Total Value of C's	# of passes	Number of C's on final pass	Total Value of C's
1	1	14	84,414	1	12	26,526
2	1	284	367,715	1	270	381,058
3	5	427	628,785	12	1440	6.83E+7
4	5	167	358,217	11	507	4.44E+7
5	1	219	310,796	1	187	242,256

TABLE 4 Closeness of P to midpoint of protection interval
Fixed Interval vs. Sliding Protection

	Fixed Interval Protection		Sliding Protection			
Data Table	Number of P's	% of P's close to midpoint	Number of P's	% of P's close to midpoint	Number of P's not receiving fixed prot.	% of P's (prev. col.) close to endpoint
1	4	0%	4	0%	2	100%
2	42	19%	42	21%	10	20%
3	44	9%	44	0%	3	100%
4	9	0%	9	0%	1	100%
5	33	27%	33	21%	5	40%

6. Conclusions

If the Census Bureau continues to feel that the value capmult=1 is a good reflection of table user knowledge of respondents' values, and that it should continue to be incorporated into the production cell suppression programs, then FP is equivalent to SP and there is no reason to implement SP. If the Census Bureau feels that other values of capmult should be explored, then further testing of SP vs. FP with additional tables would probably make sense. If capmult is set to a very large value, we could use the results of KGA together with our own results, to see what the impact would be.

Acknowledgments

I would like to thank Laura Zayatz for suggesting this problem and Jim Fagan for much helpful discussion about how to program the SP model.

7. References

[J, 1993] Jewett, Robert, "Disclosure Analysis for the 1992 Economic Census", Economic Programming Report, U.S. Census Bureau, unpublished

[M, 2001] Massell, Paul B., "Cell Suppression and Audit Programs used for Economic Magnitude Data", Statistical Research Division Report, U.S. Census Bureau, http://www.census.gov/srd/papers/pdf/rr2001-01.pdf

[KGA, 1992] Kelly, James P., Bruce L. Golden, Arjang A. Assad, "Cell Suppression: Disclosure Protection for Sensitive Tabular Data", Networks, Vol. 22, (1992), pp. 397-417

[KGA, 1990] Kelly, James P., Bruce L. Golden, Arjang A. Assad, "Cell Suppression Using Sliding Protection Ranges", Working Paper Series MS/S 90-007, University of Maryland (1990)

APPENDIX: Mathematical aspects of protection models; programming details

Setup: Assume the protection is done sequentially; i.e. protecting one primary cell P at a time; in descending order of required protection for the primaries. Suppose a certain P is ready for protection processing. Assume its one-sided protection, denoted 'prot', has been calculated using an acceptable rule, such as the p% rule. Note that in general, 'prot' depends on the microdata underlying the cell P; specifically 'prot' is a function of the distribution of the cell contributions.

The use of 'prot' for each of type of protection

1. For fixed interval protection (denoted FP), the goal is to ensure that the best estimate any table

user (other than contributor A) can derive for contributor A's value (denoted x1), is an interval U that contains the interval [x1*(1-p/100) , x1*(1+p/100)]. The value 'prot' is used to construct U as follows.
(i) Create an upper protection flow (through the table) that perturbs the (original) value of the P cell upward by 'prot'.
(ii) Create, independently, a lower protection flow that perturbs the value of the P cell downward by 'prot'.

2. For sliding protection (denoted SP), the goal is somewhat less demanding than for FP. We need only find an interval U that contains x1 and has a width of at least 2* prot; [2*prot is at most (2*p/100)*x1; this occurs when there are only 1 or 2 establishments]. There is no requirement on the percentage of this width that must lie to the right or to the left of x1.

The value 'prot' is used to construct U as follows. Simultaneously, create both upper and lower protection flows (through the table) that satisfy all the constraints of the FP model plus the following 'coupling' constraint:

Upper perturbation + Lower perturbation = (2*prot)

Flow properties

Let us consider the FP case in which the upper and lower protection flows are independent (such flows may be called 'decoupled' in the language of dynamical systems theory).

For a single flow in which cells have only the two variables, xp(i) and xm(i), (xp=upwards perturbation; xm=downwards perturbation) in each cell i, it is well known that at most one of these two variables will be positive regardless of the form of the cost function being minimized. This is a property of the simplex algorithm and network flow algorithms. It is based on the fact that these algorithms always select a set of columns from the constraint matrix that form a basis. A variable can be non-zero only if its column is in the basis selected at the end of the procedure. However, the columns for xm and xp are negatives of each other; therefore these columns can never both be in the final basis. Therefore, at most one of xm and xp can be positive in the final solution.

Let the sensitive cell (i.e., P cell) have index jp. Then the upper protection flow is a flow that is driven by requiring the conditions xp(jp)=prot and xm(jp)=0. Likewise, lower protection flow is a flow that is driven by xm(jp)=prot and xp(jp)=0.

Definition:
We say the flow passes through cell j if at least one of {xp(j), xm(j)} is positive. Note that, because of the constraints for the LP model, for some of the cells traversed (or passed through) xm(i) > 0 and for others xp(i) > 0. In other words, it is either very rare or impossible for a flow for the marginals LP model to involve only upward perturbations (or only downwards ones).

Reversibility results

Definition: A flow is reversible if the values of xp(i) and xm(i) can be exchanged for all i, and the resulting flow satisfies the model constraints.

Under what conditions can a flow be reversed ?

Let's answer this question for marginal LP models. In these models, the bounds are:

ub(xp(i)) = capmult * capacity(i)
ub(xm(i)) = capacity(i)

Case: capmult=1

Let Sp(flow) = set of cells i for which xp(i) > 0
Sm(flow) = set of cells i for which xm(i) > 0

Reversal of flow is possible since reversal of flow involves the following reassignment:

If xp(i) lies in Sp and xp(i)=f(i); for reverse flow set xm(i)=f(i) and xp(i)=0.

Likewise if xm(i) lies in Sm and xm(i)=f(i), for reverse flow set xp(i)=f(i) and xm(i)=0.

Thus we have:

Sp(reverse flow) = Sm(given flow)
Sm(reverse flow) = Sp(given flow)

Case: capmult > 1

Any flow that exploits the fact that capmult > 1 will not be reversible. By "exploits" we mean a flow in which for at least one i, xp(i) > capacity(i).

Recall that with the standard bounds for the marginal LP model, xm(i) cannot be greater than capacity(i). Thus the flow is not reversible.

A Fuzzy Result
A "nearly reversible" flow is usually possible if 'prot' is a small percentage k of value(P) and most of the cells have capacity at least as large as 'prot'.

Question: when will the upper protection (denoted UP) and lower protection (denoted LP) flows be reverses of one another ?

Answer: Reversibility is guaranteed only if capmult=1.

Case: capmult > 1

Note: If capmult > 1, it's possible that one could reverse the optimal UP flow to get a possibly not optimal LP flow, but could not reverse the optimal LP flow to get the optimal UP flow. (The reverse of an UP flow might not be optimal if for some cell i in the UP flow, xm(i) = capacity(i).)

Thm. For case capmult > 1, to get the optimal UP and LP flows, we need to solve for 2 independent flows. We could create a large LP model and solve for them simultaneously but we could just as easily solve for them sequentially since these flows are de-coupled for the FP problem.

Review:
There is a basic problem when one uses an LP model to solve the optimization problem associated with cell suppression. This is because the latter is inherently an integer programming (IP) problem. If one wants to use an LP heuristic (i.e., an approximation to the IP model) in order to get a faster running program, one needs to work with a cost function that is only an approximation to the IP model cost function.

In the IP model, the cost function CostIP= Sum (over j) of c(j) * I(i) where I(i), an indicator function, equals one only for cells within non-zero flow (is it zero otherwise). In the LP model, the cost function CostLP = Sum {over i} of c(i) * x(i), where x(i) is the flow through cell i. If one somehow knew a priori that the flows were likely to be equal (or nearly so) through all cells with non-zero flow, then x(i) could be treated as a constant. In that case, minimizing CostLP would be (nearly) equivalent to minimizing CostIP.