A Stochastic Search Approach to Solving the Cell Suppression Problem in 3-Dimensional Hierarchical Tables

Matt Fetter
United States Department of Agriculture, National Agricultural Statistics Service,
1400 Independence Avenue, SW, Washington, D.C., 20250

Abstract
Cell suppression is one method that is commonly used to reduce disclosure risk when data are published in hierarchical tables. A form of optimality is achieved for 2-dimensional tables by formulating the cell suppression problem as a minimum cost flow problem. There are issues with this approach in general, and for its application to 3-dimensional tables in particular. First, cell suppression is fundamentally an integer programming problem with a non-smooth cost function. Secondly, the minimum cost flow approach is not directly applicable to 3-dimensional tables. A stochastic search approach is presented that is guaranteed to generate closed paths in 3-dimensional tables. Although no claim of optimality can be made, this method is capable of finding good solutions with respect to the non-smooth cost function associated with the cell suppression problem.

Key Words: stochastic search, statistical disclosure control, cell suppression.

Introduction

Cell suppression is a statistical disclosure control methodology commonly employed when data are published in tabular format. In theory, these tables can be of any dimension >=1. The focus of this paper is on tables containing magnitude data that are of three dimensions. These tables have three marginal totals- the row sums, the column sums and the level sums. Only simple tables with no sub-totals in any of the margins are considered; tables with sub-totals in one of the margins can be thought of as a linked simple sub-table of the original table and treated accordingly. Further development will be necessary to handle these more complicated tables- although the core algorithm should still be directly applicable without modification. The algorithm will be referred to as Row Column Level EXchanger or “R-CLEX”.

Each dimension of the table is defined by a set of specific characteristics called explanatory variables. The axis of each explanatory variable is divided up into intervals that demarcate different ranges of values for that particular variable. Table cells are then formed by the intersection of these intervals. The location of each cell in the table can be identified by a unique 3-tuple, corresponding to the interval index for each of the 3 dimensions.

Typically, some survey weight is applied to the respondent data in the cell and summed to give some estimate of the population total for the portion of the target population that would comprise that particular cell. Marginal totals are obtained by holding two of the three dimensions fixed and then summing the individual cell totals across the third.

Due to the distribution of the respondent data within a cell, it is sometimes possible for a person analyzing the data to determine a narrow range of possible values for a particular respondent in the cell. If this range is too narrow, the creator of the table will deem this as a sensitive cell and will require the contents of the cell to be suppressed. Instead of the aggregated total, the published table will simply have a suppression indicator, i.e., a ‘D’ in the sensitive cell. However, it is generally not sufficient to suppress only the sensitive cell. This is because table rows, columns, and levels will sum to marginal totals that are also published.

As an example, consider a 1-dimensional table where the marginal total for a characteristic defining the single dimension of the table is T. The individual cell totals defined by specific values for this characteristic, say t1, t2, and t3 will sum to T. Algebraically this is written as T= t1+t2+t3. If t2 is a sensitive cell, it must be suppressed. We replace t2 with a ‘D’ and write T=t1+D+t3. Clearly, D=T-t1-t3, so the value in t2 can still be exactly determined. No protection is afforded the sensitive cell by simply suppressing the sensitive cell. It is therefore necessary to select an additional cell to suppress. This suppressed cell is referred to as a “complimentary” suppression. Determining which cells to choose as complimentary suppressions while maintaining a constrained optimality condition constitutes the cell suppression problem (CSP).

There are different methods that can be used to solve CSPs. The approach used depends on how one chooses to model the CSP and the characteristics of the tables involved. For 2-dimensional tables, a Linear Programming (LP) approach is often taken. More sophisticated methods model the CSP as an Integer Programming problem (IP). Some methods can handle 2-dimensional but not 3-dimensional tables while other methods can handle tables containing relatively few cells but not large tables with tens of thousands of cells.

A detailed explanation of the theory of how solving the CSP can be used to reduce the risk of disclosure for tabular data is well known and will not be described in this paper. Rather, this paper describes a new approach to finding good solutions for 3-dimensional CSPs using a stochastic search method. The CSP itself can be understood and appreciated outside the context of statistical disclosure control without any knowledge of how solving it protects sensitive cells. It is simply an optimization problem with a discrete cost function and a set of constraints.

The Model

The 3-dimensional CSP model under discussion will be expressed mathematically as:

$$\boldsymbol{min}\, L(\theta) = \theta^T (w \circ X + a) \qquad (1)$$

$$s.t.: \theta^T \theta = \text{the specified path size} \qquad (2)$$

$$\text{if } q_{crl} = 1 \text{ then :}$$

$$\sum_c q_{crl} = 2, \sum_r q_{crl} = 2, \sum_l q_{crl} = 2 \qquad (3)$$

$$\text{and } x_{crl_n} \geq d, \text{ a specified positive constant} \qquad (4)$$

Where:

$$X = \left[x_{crl_1}, x_{crl_2}, \ldots\ldots, x_{crl_N}\right]^T \qquad (5)$$

$$x_{crl_n} = \text{value of cell } n \text{ located at } (c, r, l)$$

$$w = \left[w_{crl_1}, w_{crl_2}, \ldots.. w_{crl_N}\right]^T \text{ (a column vector of weights)} \qquad (6)$$

$$a = \left[a_{crl_1}, a_{crl_2}, \ldots\ldots. a_{crl_N}\right]^T \text{ (a column vector of per - cell penalty costs)} \qquad (7)$$

$$\theta = \left[q_{crl_1}, q_{crl_2}, \ldots\ldots.., q_{crl_N}\right]^T \text{ (a column vector of suppression indicators)} \qquad (8)$$

$c =$ column

$r =$ row

$l =$ level

$N =$ number of cells in table

$q_{crl_n} =$ suppression indicator for cell n located in column c, row r, level l

$q_{crl_n} = 1$ if cell n is included in the path (suppressed)

$= 0$ otherwise

Equation (1) gives the cost function. It is the weighted sum of the suppressed cell values, plus an arbitrary base per-cell penalty cost. In the most basic set up, the weights in (6) are set to 1 and the penalty costs in (7) are set to zero for all cells in the table. The vector of indicator variables, θ, ((2) and again in (8)), defines the decision vector. Each component of this vector is linked to a unique cell in the table (5). If the cell in a table needs to be suppressed, its corresponding component in the decision vector is set to one, otherwise it is set to zero. If there are N cells in the table, the decision vector contains N components. There are 2^N distinct possible configurations of this decision vector. Fortunately, most of them are of no interest because they are not feasible. They are not feasible because they violate side constraints associated with the CSP.

For 2-dimensional tables, the smallest feasible solution would require suppressing the sensitive cell and three complimentary cells. This eliminates all configurations of the decision vector that have fewer than four components set to one. For 3-dimensional tables, seven complimentary cells are required to protect a sensitive cell thus requiring a total of eight suppressed cells. Furthermore, feasible solutions are limited to specific numbers of cells. In three dimensions, feasible solutions must have a minimum of eight suppressed cells. Larger solutions must contain an even number of cells, excluding 10 (this might be an interesting geometric or number theory problem). Feasible solutions that involve a large number of cells are likely to be of little or no interest.

The constraints placed upon the solution to the CSP are concerned with either the number of suppressed cells in the feasible solutions allowed (2), or their relative positions in the table (3). Equations in (3) define a "closed path" and are the key to assuring that the exact value of the sensitive cell can't be obtained by applying a series of subtraction operations using the marginal totals. The positive constant, *d,* in (4) relates to the amount of protection required for the sensitive cell. It assures that the range of possible values that can be determined for the sensitive cell is sufficiently wide.

All solutions obtained by R-CLEX will be closed paths because it considers the constraints given in (3) to be mandatory. This set of constraints is the single most critical aspect concerning protection of sensitive cells. R-CLEX is not forced to adhere to the protection constraint given in (4), but solutions that violate this constraint are heavily penalized in terms of cost. This makes it unlikely that R-CLEX will produce such solutions.

The CSP in the expression given in (1) represents an integer programming problem. This problem is considered a strongly NP-hard problem, meaning that it is unlikely that any algorithm could be developed that would be able to find the optimum solution in any reasonable amount of time in all cases. This implies that solutions obtained by R-CLEX will most likely not be optimal except for tables containing sufficiently few cells. Given this fact, the idea is to find good (i.e., reasonably inexpensive), feasible solutions using a method that protects against the possibilities of getting really expensive or infeasible solutions.

R-CLEX is a stochastic search method. Stochastic searches are non-deterministic because they utilize a random component to how the search is conducted. Depending on the problem at hand, different runs of the algorithm can produce different solutions. One of the strengths of stochastic methods is that they can often find good solutions to problems that are very difficult or impossible for deterministic analytical methods to solve. In particular, optimization problems that involve discrete cost functions such as the CSP described in (1) fall in this category.

The CSP problem as stated above considers only a single sensitive cell. In practice, a given table is likely to have many sensitive cells. Simply finding the optimal solution for each sensitive cell separately does not generally result in minimizing the total cost across all suppressions in the table. It would be ideal to optimize for all sensitive cells in the table *simultaneously*. This is a much more difficult problem for R-CLEX (and many other methods as well).

R-CLEX does employ a heuristic to help it to find a good simultaneous solution across all sensitive cells. R-CLEX deals with a table one sensitive cell at a time. A queue is formed based on each sensitive cell's required protection. The cells with the largest required protection are handled first; cells with the least required protection are handled last. After a solution is determined for the first cell, solutions for all subsequent cells are conditional on the set of cells that are already suppressed. This technique increases the likelihood that many of the suppressed cells will be used to help protect multiple sensitive cells simultaneously. This can result in reducing the overall cost at the table level.

R-CLEX searches for good solutions by exploiting the geometric requirements of any closed path in three dimensions. Figure 1 shows what a typical 8-cell closed path in three dimensions might look like.

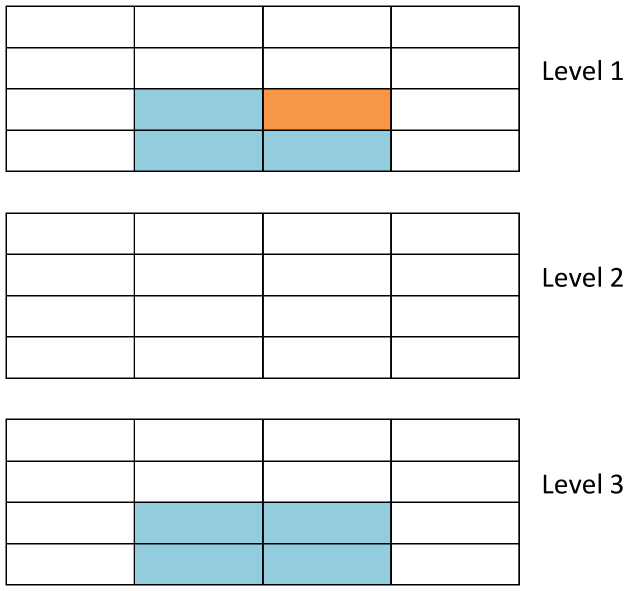


Figure 1.

This table has 4 rows, 4 columns and 3 levels. The orange cell represents the sensitive cell. The blue cells represent the complementary cells. The shaded cells taken together represent the closed path. Note how the constraints expressed in (3) would be satisfied by this path.

Figure 2 shows a 12-cell closed path.

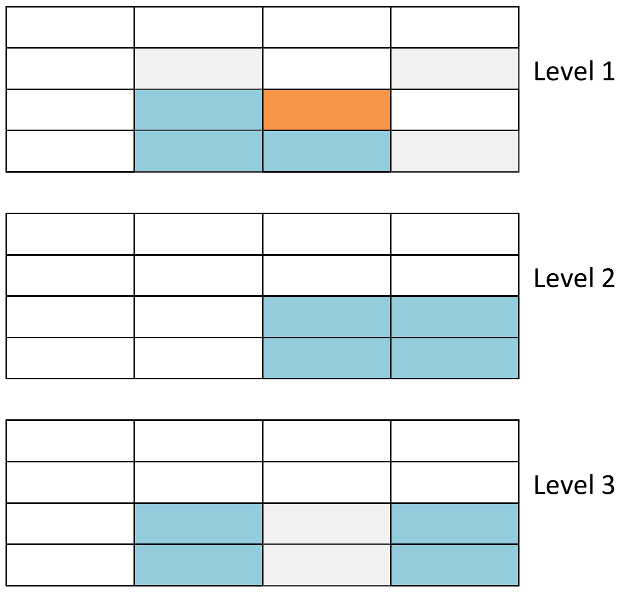


Figure 2.

Again, the constraints in (3) would be satisfied.

The paths can get quite complicated as shown in Figure 3, depicting a 24-cell closed path in 3 dimensions.

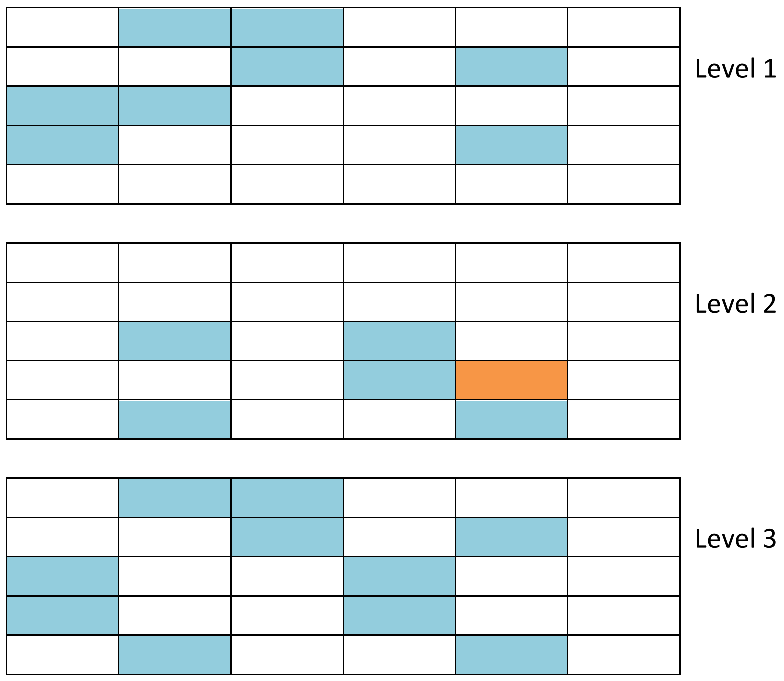


Figure 3.

Beginning at any suppressed cell, holding any two of the coordinates fixed and counting across the third will result in exactly two suppressed cells in all cases. This defines a closed path.

In the author's limited experience, the 8-cell path is the most important. It tends to be cheaper simply because it contains fewer cells thus returning lower costs. However, typically, a table will have many sensitive cells. It is sometimes the case that sensitive cells can help protect each other if they are included in the same path. In these situations, the larger more complicated paths can sometimes be an advantage in helping to reduce the over-all cost of the suppressions across all sensitive cells.

It is interesting to note that the total number of 3-dimensional 8-cell closed paths in a 3-dimensional table is relatively small for a single sensitive cell. The number of such paths given by the following formula:

$$P_{3D\text{-}8} = (R-1)*(C-1)*(L-1) < N,$$

where N represents the number of cells in the table, and C, R, and L represents the number of table columns, rows, and levels, respectively.

This relation implies that a systematic exhaustive search of all possible 8-cell paths can be manageable until the table gets fairly large. (R-CLEX does not currently do a systematic search.) A 10 x 20 x 100 table has only about 17,000 such paths for a given sensitive cell, probably within the capabilities of a desktop computer for dealing with a few sensitive cells. However, the number of closed 3-dimensional paths containing as few as 12 cells explodes and exhaustive systematic search becomes much more complicated and prohibitively time consuming. The number of such paths for a 10x20x100 cell table would be over 40 million. The following formula gives the number of 12-cell 3-dimensional paths in a table for a given sensitive cell.

$$P_{3D\text{-}12} = (R-1)*(R-2)*(C-1)*(C-2)*(L-1) + (R-1)*(R-2)*(L-1)*(L-2)*(C-1) + (C-1)*(C-2)*(L-1)*(L-2)*(R-1).$$[1]

Detail Concerning Closed Path Geometry

The *kernel* of a path denotes the number of distinct values of the three coordinates that occur in the path. All 8-cell 3-dimensional closed paths have the kernel *K(2,2,2).* This means that each of the coordinates have two distinct values. All 12-cell paths have the kernel *K(3,3,2)*. This means that all 12-cell paths have two coordinates with 3 distinct values and one coordinate with two distinct values. The kernel does not imply order i.e., *K(3,3,2)≡K(2,3,3).* Larger paths can have multiple kernels for a path of a given size. Paths containing 16 cells can be of one of two kernels, *K(4,4,2)* or *K(4,3,2).*

[1] This formula assumes R>=3, C>=3, L>=3.

The *configuration* of a path gives the *ordered* distinct values for the coordinates in the path. All 8-cell paths have a configuration of *C(2,2,2)*. However, 12-cell paths can have one of three configurations; *C(2,3,3)*, *C(3,2,3)*, or *C(3,3,2)*. The 12-cell path shown below in Figure 4 has a configuration of *C(3,3,2)* since there are three distinct columns in the path, three distinct rows, and two distinct levels.

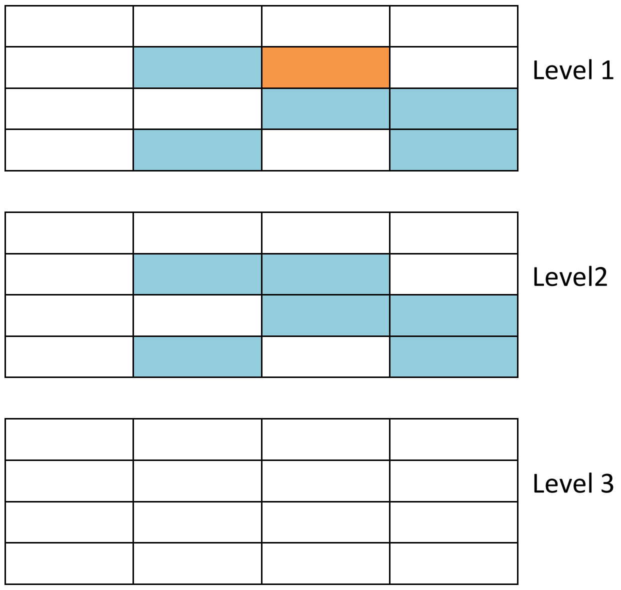


Figure 4.

A 12-cell path with *C(2,3,3)* is shown below in Figure 5.:

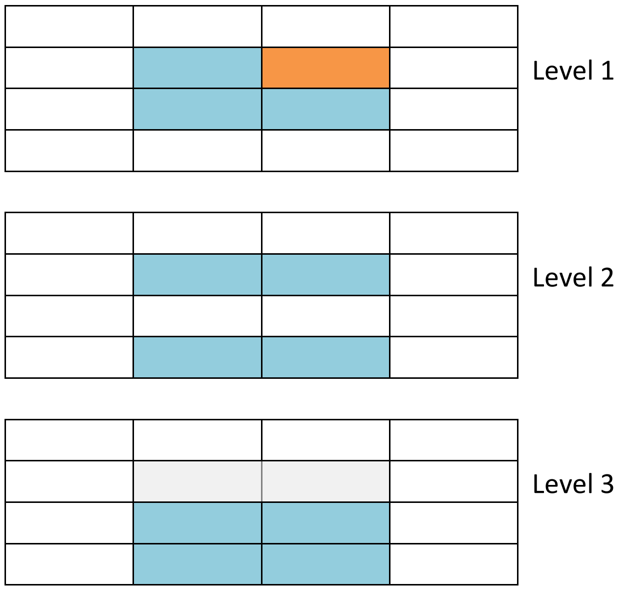


Figure 5.

The concepts of kernel and configuration have important theoretical and algorithmic implications for how R-CLEX operates. The key to how R-CLEX works is that any 3-dimensional closed path of any kernel or configuration is comprised of a set of distinct components. These components are the columns, rows, and levels that are involved with the candidate solution path. In Figure 5, columns 3 and 4 are involved with this path. Likewise, rows 2, 3, and 4 as well as levels 1, 2, and 3 are involved with the path.

Every iteration of R-CLEX produces a candidate solution. The user specifies the number of iterations that R-CLEX is to execute. Once the budget of iterations is exhausted, the least expensive candidate solution is taken as the operational solution for protecting the sensitive cell being serviced. At each iteration, R-CLEX will try to improve upon the current solution by first randomly choosing to make either a row, column or level exchange. This decision is made with equal probability given to each of the three possible choices. Components that are involved with the current candidate solution will be referred to as *active.* All other rows, columns or levels in the table will be referred to as *inactive*

If R-CLEX chooses to do a row exchange, for example, then R-CLEX will evaluate each active row's contribution to the total cost. Each row's cost is computed by holding the row fixed and summing the cell values across all suppressed cells possessing that row coordinate. The row containing the sensitive cell is not eligible for selection. The idea is to select with a high probability (but less than certainty) the active row with the highest cost and release it from the current solution path. An active row that contains suppressed cell(s) with a value less than the protection requirement, *d*, is given a very high cost. Such a row, if it exists, is very likely to be chosen for release. In a similar manner, R-CLEX will select (also with a high probability but less than certainty) the cheapest inactive row as a replacement. Inactive rows that contain cells with values less than *d* that are in the necessary positions are given a very high cost and are much less likely to be chosen as the replacement row. In short, R-CLEX attempts to shed expensive components and replace them with cheaper components as it iterates.

These exchanges are done in a way that insures that the new candidate solution adheres to the closed path constraint and also has a high probability of meeting the protection constraint. A closed path is maintained for a row exchange by simply changing the row coordinate of each cell in the selected row of the current path to the row coordinate of the replacement row, keeping the other two coordinates fixed. Column and level exchanges are accomplished in a similar manner, fixing the row and level coordinates, or the row and column coordinates, respectively.

The next figures show a row exchange, releasing row 3 in Figure 6a, and replacing it with row 1 in Figure 6b.

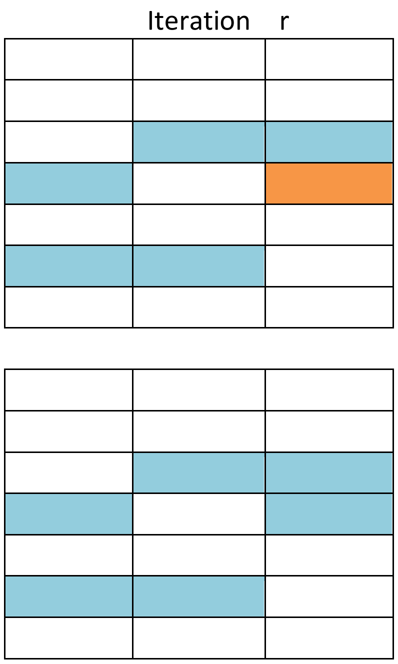


Figure 6a.

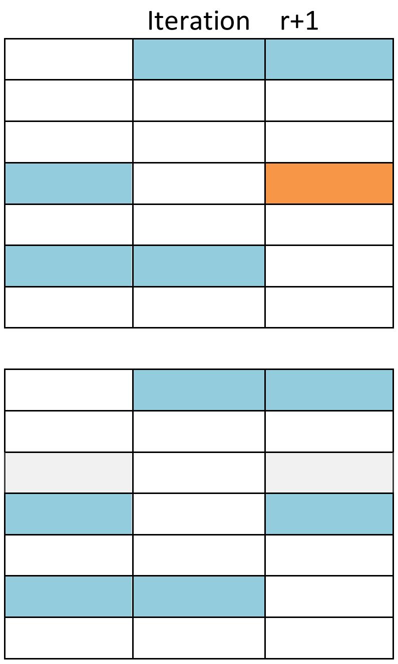


Figure 6b.

It is important that R-CLEX has the potential to explore the entire space of feasible solutions of a given path size. Row, column, and level exchanges are sufficient to explore the entire space of 8-cell solutions because all such solutions have the same kernel and configuration. However, solutions with larger path sizes can be found in more than one configuration. Row, column and level exchanges alone will never allow solutions to move across configurations. Configuration changes require coordinate scrambling. This is easily accomplished and can best be explained by an example.

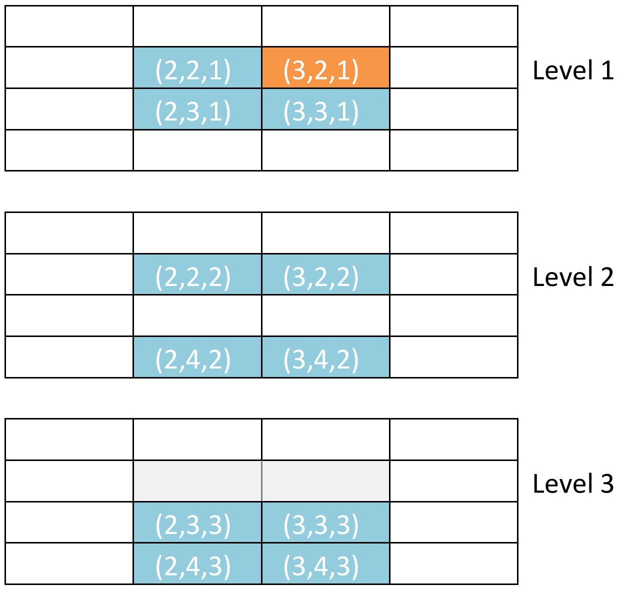


Figure 7.

The path shown above in Figure 7 is a *C(2,3,3)* 12-cell path. This path can be translated to a *C(3,2,3)* 12-cell path by switching the row and column coordinates for each cell in the path, resulting in a solution that looks like this:

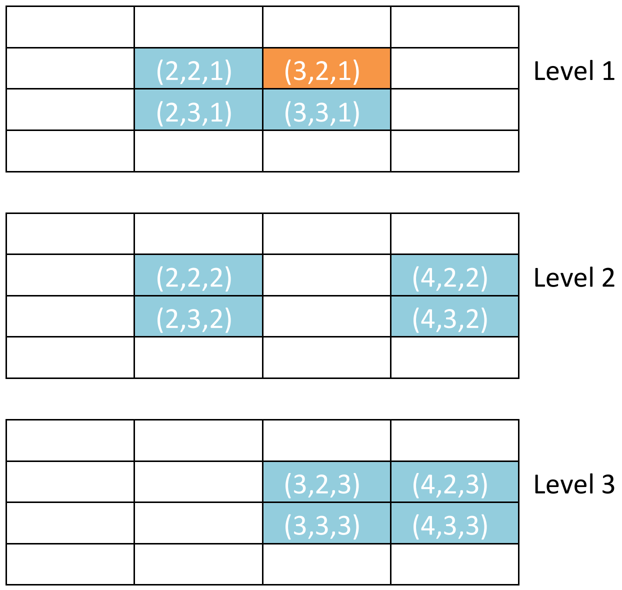


Figure 8.

The resulting solution is *C(3,2,3)* as desired.

Many solution paths that are larger than 12-cells, i.e., 16-cell paths, have multiple kernels as well as multiple configurations for each kernel. Coordinate scrambling will not translate a 16-cell path of *K(4,4,2)* to one of *K(4,3,2)*. If a user wants to consider all 16-cell paths, R-CLEX must be initialized with a 16-cell path of each kernel.

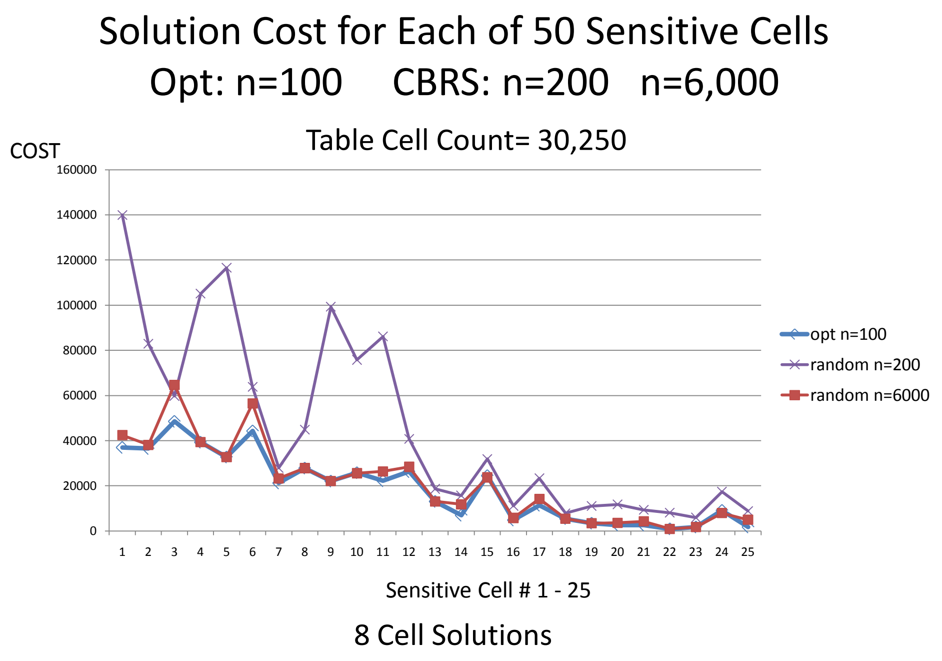
Currently, R-CLEX is initialized with two arbitrary closed paths of user specified lengths and kernels. The lengths of these paths can be the same or different, depending on what the user desires. These initial paths are supplied by a permanent library of closed paths with path lengths ranging from 8 to as many as 38 cells, with multiple kernels available for each path length. The library is produced by a different algorithm that generates 3-dimensional closed paths on the fly. Once the library is created, it can be stored on disk and accessed as needed.

Experience has shown that initial paths of 8 and 12 cells seem to be good choices for finding good solutions. R-CLEX produces two candidate solutions at every iteration, one for each path size/kernel specified. If the two initial path lengths are the same, two candidate solutions are still produced, both the same length. In either case, the path that is cheaper is retained as the iteration's representative candidate solution and the more expensive solution is discarded. In this way, R-CLEX can explore paths of different lengths simultaneously when searching for solutions.

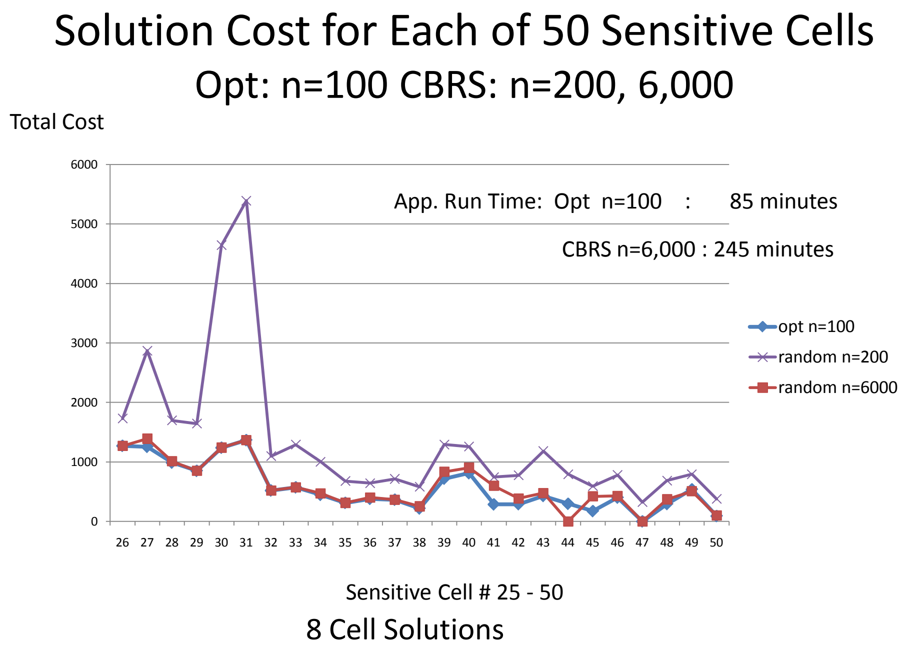
A Few Simulation Results

An artificial 121 x 25 x 10 table was populated with zeroes and positive integer values. Approximately 15% of the cells contained a zero value. Fifty cells were chosen to represent sensitive cells. These were scattered around the table. Assigned protection requirements ranged from *d* =2 to *d*= 3,000 units. Recall the constraint given by (4) requires that the cell values in the path must at least equal *d*. In general, this causes paths protecting cells that require greater protection to be more expensive. The cost function used in the evaluations is given in (1) with weights, *w*, set to 1 and the per-cell penalty, *a*, set to zero for all cells.

The performance of R-CLEX is compared to a constrained, blind random search (CBRS). The CBRS is performed by R-CLEX, but with the R-CLEX "optimizing" strategy disabled. This means that CBRS is still a fairly sophisticated and possibly useful method to solve the CSP. It will always produce closed path solutions. By disabling the "optimizer", CBRS takes less time per iteration. This is because CBRS omits the cost analysis step that is required by the R-CLEX "optimizer". CBRS makes all search decisions randomly with equal probability. It makes no special effort to release the most expensive component from the current candidate solution, and makes no special effort to find the least expensive, inactive component to use as a replacement. This comparison gives some insight into the effectiveness of the optimization strategy used by R-CLEX. One would expect CBRS to take more iterations than R-CLEX to obtain comparable solutions. The main interest is to determine if R-CLEX is significantly faster than CBRS in terms of wall clock run time.



Graph 1a.



Graph 1b.

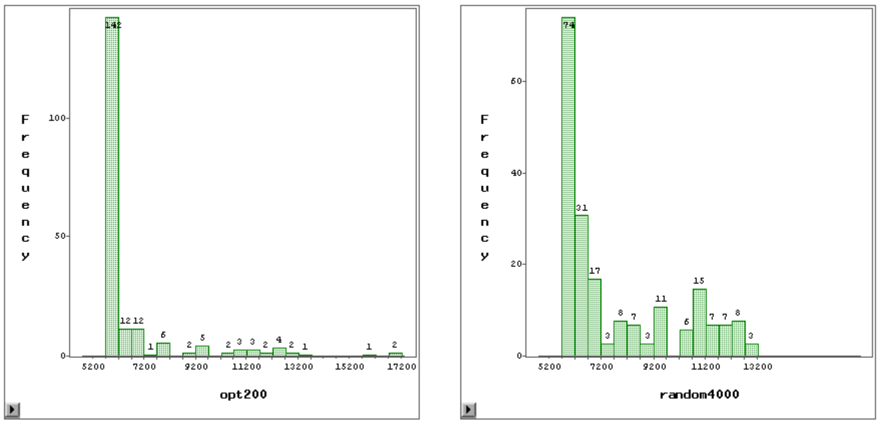
By examining Graph 1a and Graph 1b above, one can see that in general, R-CLEX (Opt) requires just a small fraction of the iterations, and about one-third the time CBRS (random) uses to obtain comparable solutions for this set of problems. The relative performance of R-CLEX to CBRS will vary depending on the particular problem being investigated as will be seen in the next test problem.

In the next test, the qualities of the solutions that are produced by R-CLEX are examined. A 121 x 25 x 10 table was set up and then populated with zero cells. A very small number (less than 20) of these zero cells were changed to positive values in such a way as to define a sensitive cell and a few 12-cell closed paths that could be used to sufficiently protect it. This is a “find the needle in the haystack” type of problem. R-CLEX had no trouble solving this problem consistently giving full protection using only 100 iterations. CBRS was never successful at finding a path that did not contain some zero cells (meaning the path failed to provide sufficient protection to the sensitive cell), even after 100,000 iterations. In some ways, this is not surprising given that there are millions of 12-cell paths in this table and only the tiniest fraction of them are any good.

In another test, a 10 x 10 x 4 table was created and populated with positive integer values. One cell was designated as the sensitive cell requiring an arbitrary (but sensible) amount of protection. Some limited testing of the quality of R-CLEX solutions is obtained by examining the distribution of solutions obtained by 200 independent 200 iteration runs and comparing this distribution to the known optimal solution of 5,742. This can be compared to results obtained by CBRS using 200 independent 4,000 iteration runs.

BEST SOLUTION DISTRIBUTION- Single Sensitive Cell

Mean Cost: 6,864 12 Cell Solutions Mean Cost: 7,827



Optimal: n=200 CBRS: n=4,000

Single Sensitive Cell: 12 cell Minimum Cost: 5,742

Graph 2.

Graph 2 gives the cost (horizontal axis) frequency distribution with R-CLEX on the left and CBRS on the right. Even the most expensive solutions displayed in Graph 2 are very inexpensive compared to the cost of the large majority of other paths that could be found in the table.

Concluding Remarks

Today, there are many alternatives to using cell suppression for the management of disclosure risk in tabular data. Controlled tabular adjustment (CTA), various perturbation methods, and synthetic data are some examples. However, cell suppression is still a standard method for limiting disclosure risk for tabular data.

There are numerous other strategies available for solving CSPs in three or more dimensions such as one developed by Fischetti and Salazar-Gonzalez (2001). This powerful and mathematically sophisticated method is implemented in the software package, tau-ARGUS.

R-CLEX provides an interesting example of how stochastic search can be applied successfully to the CSP. It is the only stochastic search method known to the author that can find CSP solutions of any feasible path length. The method is simple and straight forward. It lends itself to parallel processing for increasing speed and enhancing performance. A mathematical formulation of the geometric qualities of a closed path is incorporated into the constraints of the model for which R-CLEX obtains solutions. The general geometric properties of 3-dimensional closed paths are developed and are at the heart of the algorithm.

R-CLEX does not create closed paths on the fly (this seems to be a pretty tricky- and relatively inefficient approach) but is initialized with a closed path obtained from a static library of closed paths with desired size and kernel. This library is produced by a one-time run of an accompanying heuristic-based generator algorithm that creates closed paths from scratch. Paths containing 8, 12, and 14 cells have only one kernel, and initial paths with these lengths can easily be created "by hand". Larger paths will probably have multiple kernels. The generator program is recommended to create the larger paths for initializing R-CLEX.

R-CLEX and the generator algorithm are written in SAS® using the IML procedure and can run on a desktop computer. It is not yet, however, a finished product. Only simple hierarchical tables without row sub-totals can currently be handled by R-CLEX, although this is a limitation of the infrastructure surrounding the R-CLEX core algorithm and not the algorithm itself. Further development for handling more complicated 3-dimensional tables is ongoing.

References

Cox, L.H. (1995). Protecting confidentiality in business surveys. In *Business Survey Methods* Edited by: Cox, B.G., Binder, D.A., Chinnappa, B.N., Christianson, A., Colledge, M.J., Kott, P.S. (pp.443-473). New York, NY: John Wiley and Sons, Inc.

Fischetti, M. and Salazar-Gonzalez, J. J. (2001). Models and algorithms for optimizing cell suppression in tabular data with constraints. *Management Science*, 47(7), 1008-1027.

Massell, P. B. (2002). Optimization models and programs for cell suppression in statistical tables. *Joint Statistical Meetings*.

Spall, J.C. (2003). *Introduction to stochastic search and optimization:- estimation, simulation, and control*. Hoboken, NJ: John Wiley and Sons, Inc.