Integrated Exact, Hybrid and Metaheuristic Learning Methods for Confidentiality Protection

Fred Glover
OptTek Systems, Inc.; Boulder; CO; Glover@OptTek.com

Lawrence H. Cox

National Center for Health Statistics, Centers for Disease Control and Prevention, Hyattsville, MD; LCox@CDC.gov

Rahul Patil
SJMSOM, Indian Institute of Technology, Bombay, India; rahul.patil@iitb.ac.in, ph: 091-22-25767784

James P. Kelly
OptTek Systems, Inc.; Boulder, CO; Kelly@OptTek.com

Abstract

A vital task facing government agencies and commercial organizations that report data is to represent the data in a meaningful way and simultaneously to protect the confidentiality of critical components of this data. The challenge is to organize and disseminate data in a form that prevents such critical components from being inferred by groups bent on corporate espionage, to gain competitive advantages, or having a desire to penetrate the security of the information underlying the data. Controlled tabular adjustment is a recently developed approach for protecting sensitive information by imposing a special form of statistical disclosure limitation on tabular data. The underlying model gives rise to a mixed integer linear programming problem involving both continuous and discrete (zero-one) variables. We develop stratified ordered (s-ordered) heuristics and a new meta-heuristic learning approach for solving this model, and compare their performance to previous heuristics and to an exact algorithm embodied in the state-of-the-art ILOG- CPLEX software. Our new approaches are based on partitioning the problem into its discrete and continuous components, first creating an s-ordered heuristic that reduces the number of binary variables through a grouping procedure that combines an exact mathematical programming model with constructive heuristics. To gain further advantages we then replace the mathematical programming model with an evolutionary scatter search approach that makes it possible to extend the method to large problems with over 9000 entries. Finally, we introduce a new metaheuristic learning method that significantly improves the quality of solutions obtained.

Keywords: *confidentiality, mixed integer optimization, metaheuristics, adaptive learning, mathematical programming, evolutionary computation.*

An earlier, incomplete, and abbreviated version of this paper appears with the title "Exact,Heuristic and Metaheuristic Methods for Confidentiality Protection by Controlled Tabular Adjustment," *International Journal of Operations Research*, Vol. 5, No. 2, pp. 117-128, 2008.

1 Introduction

Data confidentiality is an essential component of statistical surveys, for three reasons. The first involves laws or government regulations pertaining to a particular survey (or to the agency collecting the data) that require responses provided to the survey be held confidential by the agency. Such laws can be very broad, as for the Australian statistical system, that in essence require due diligence. They can alternatively be very specific, as for the U.S. Internal Revenue Service, requiring that "fact of filing" not be divulged and prescribing a "rule of three" for aggregations, or for the U.S. Census Bureau, requiring that "no information provided under (this act) be divulged" to a third party. Recently, the U.S. Confidential Information Protection and Statistical Efficiency Act (CIPSEA) has imposed a blanket confidentiality protection requirement on all U.S. statistical data collections. This Act does not supplant existing stronger individual acts, but rather assures uniform minimum level of protection across agencies.

The second reason is ethical. Confidentiality protection is regarded as ethical statistical practice and appears specifically in the Codes of Ethical Conduct of the International Statistical Institute, the American Statistical Association, and other statistical organizations. The paradigm for statistical data collection is the "social contract" between the respondent and the data collector. In return for something of value from the respondent (confidential information), the data collector ("honest broker of information") provides assurance that the data are necessary to achieve something of value to society (and, presumably, the respondent) but in any case will hold individual data confidential.

The third reason is practical. Respondents would not respond, or at least would not do so as completely and truthfully as they might otherwise, if they did not have faith in the intent and ability of the data collector to preserve confidentiality. This is true even for surveys where response is mandatory. In essence, it is important on a practical as well as legal and ethical level for the data collector to make and keep promises of confidentiality. This trust is reflected in many U.S. government surveys where response rates exceed 80%.

Government agencies and commercial organizations that collect, store and report data typically have a responsibility to protect the confidentiality of these data. Tabular data are a staple of official government statistics, encompassing two data classifications: *count data* and *magnitude data*. Magnitude data are ubiquitous, referring to data on costs, receipts or sales, number of hospital admissions or days in hospital, tonnage of materials shipped or tons of pollutants emitted, etc. Many thousands of tables of magnitude data of economic, social and political importance are published each year and, based in whole or in part on these data, many key economic and policy decisions are made daily by governmental, business and community organizations.

Firms and individuals make use of these tables of economic and social data for a wide range of important purposes. Examples are as follows: (1) to study industry and business markets: gauge the competition, calculate market share, locate business markets, design sales territories and set sales quotas, etc; (2) to evaluate investment opportunities, as by assessing new business opportunities, enhancing business opportunity presentations, etc; (3) to support public sector initiatives, as by maintaining local tax bases, assisting local businesses, and making public policies.

The need to safeguard the confidentiality of such data presents a monumental task, and government agencies such as the U.S. Bureau of Census and the U.S. National Center for Health Statistics that regularly report such data must wrestle with the confidentiality problem on a continuing basis. Inadvertent disclosure of sensitive corporate information could injure a large commercial enterprise financially and in other ways, while at the same time government reporting agencies are duty-bound to provide figures that convey meaningful and accurate information about the state of our national economy and its component industries and sectors. Consequently, the confidentiality problem looms as a major challenge with far-reaching consequences. The continuing challenge is to maximize data quality and usability while preserving confidentiality.

The importance of the confidentiality protection problem is confounded by its computational complexity. The primary mechanisms, cell suppression, data rounding and controlled tabular adjustment, are expressed as binary decision problems subject to linear constraints involving potentially many binary variables. Moreover, reporting agencies must solve such problems on an ongoing basis in potentially many (survey) settings. Thus, the confidentiality problem for tabular data is in most cases not solvable optimally or even feasibly by standard algorithmic approaches.

The purpose of this paper is to propose a new method for solving data confidentiality problems. We demonstrate that we are able to achieve a significant advance over previous methods by building on a recently proposed modeling approach called *controlled tabular adjustment* (CTA). Our study includes an extensive empirical investigation of alternative methods for handling the underlying mixed integer/continuous optimization formulation that is derived from the CTA model. Our study compares previously proposed heuristics to the state-of-the-art ILOG-CPLEX optimizer, and then creates additional methods consisting of a hybrid approach, a combined hybrid scatter search approach and a new metaheuristic learning approach. Our computational investigations disclose the remarkable difficulty of solving the basic problem due to the inherent combinatorial complexity of effective confidentiality protection, and show how the new procedures provide advances over previous methods. Most significantly, we show that the metaheuristic learning method succeeds in improving the solutions to a degree that establishes this class of models not only as a theoretical contribution but as a truly practical advance for safeguarding sensitive information.

Model Considerations

In tabular data, a cell is considered *sensitive* if the publication of the true cell value is likely to disclose a contributor's data to the public or a competitor. For example, in an economic survey, if a cell contains data from one respondent, then publication of the cell value would disclose confidential data pertaining to a single respondent. As identities of companies associated with individual cells are publicly available, publication of cell values would breach the pledge of confidentiality made to the company by the statistical office collecting the data. Similarly, if a cell contains data from two respondents, or if the cell total is *dominated* by the contributions of two respondents, then either respondent could subtract its own contribution from the cell value to obtain a tightly bounded estimate of the other's contribution.

Confidentiality protection for tabular data is based on assuring that all released tabular cells satisfy an appropriate *disclosure rule* (Cox 1981; Cox 2001; Willenborg and Waal 1996; 2001). Cells failing to satisfy the rule (sensitive cells) are assigned protection ranges defined by lower and upper bounds on the true cell value computed from the disclosure rule. Published or derived estimates of the original values of sensitive cells that lie within the protection range are considered unacceptable.

Different procedures have been used by statistical offices to protect confidentiality of sensitive cells in tabular data. The most popular method is complementary cell suppression (Cox 1980; Kelly et al. 1992; Fischetti and Salazar 1999; 2000). The complementary cell suppression method suppresses both primary (sensitive) and secondary (nonsensitive) cells to protect the confidentiality of the sensitive cells. Although suppression is widely used, it has serious limitations. Complementary cell suppression is an NP hard problem (Kelly et al. 1992). More important, it produces tables with data missing not-at-random and therefore difficult to analyze by standard and all but the most advanced statistical methods.

To overcome the limitations of complementary cell suppression, we propose new methods that are designed to exploit the model called Controlled Tabular Adjustment (CTA) which affords an opportunity to overcome many of the problems associated with traditional cell suppression and perturbation methods. CTA introduces controlled perturbations (*adjustments*) into tabular data that satisfy the protection ranges and tabular constraints (*additivity*) while minimizing data loss as measured by one of several linear measures of overall data distortion, such as the sum of the absolute values of the individual cell value adjustments. CTA replaces each sensitive cell by either of the two endpoints of its protection range. These values are sometimes referred to as the *minimally safe values*. Selected nonsensitive cell values are then adjusted from their true values by small amounts to restore additivity. Additionally, nonsensitive cell perturbations are constrained to be small or insignificant, such as limiting them to be within sampling variability, and cell values for which adjustment is deemed undesirable can be held fixed. Cox (2000) provides a mixed integer programming formulation for CTA. Danderkar and Cox (2002) provide heuristics for solving the integer variables to be examined here.

The end result of CTA is a tabular system without suppressions meeting the disclosure rule, which is optimally close to the original system with respect to distortion measure. Thus, if the model can be solved effectively, CTA provides a safe, completely populated and fully analyzable tabular system. An additional potential advantage of the model is that the intruder is stymied by the fact that sensitive cells are not highlighted as they may appear to be under cell suppression. From the modeling standpoint, CTA is scaleable to large, multi-dimensional and complex tabular systems and provides confidentiality protection in such a way that the outside observer has no information on how the data were modified, thus reducing risk of disclosure and controlling information loss. CTA therefore offers the promise of enhancing data access while protecting confidentiality, motivating the development of an algorithmic design capable of unlocking its computational complexity.

The empirical evaluations of methods for CTA conducted in this paper begin by comparing heuristic procedures proposed by Danderkar and Cox (2002) to the commercial CPLEX solver. These analyses disclose both useful features and significant limitations of these approaches (including severe limitations to

CPLEX as problem size increases to dimensions often encountered in practice). This leads to our development and analysis of two new alternative methods embodying strategies of grouping and evolutionary scatter search, which prove more powerful than the previous heuristic approaches. Scatter search offers particular advantages by running far more efficiently than CPLEX, and significantly extending the size of problems that can be addressed, yet still encounters limitations shared with its predecessors in generating solutions of high quality. Finally, we develop a new metaheuristic learning method that performs far more effectively than all of the other methods and provides a reliable and efficient approach for producing high-quality solutions for problems of practical size.

Our development is organized as follows. Section 2 presents the mixed integer/continuous optimization model for CTA. Section 3 describes numerical tests using the heuristic approaches suggested by Danderkar and Cox (2002), including an examination of multiple objective functions and an evaluation of their impact on the CTA process. Section 4 describes a new hybrid heuristic, based on reducing the number of binary variables through grouping and combining the exact CPLEX solution approach with principles embodied in the heuristics. Section 5 introduces an evolutionary scatter search approach that replaces CPLEX and extends the method to large problems with over 9000 entries. Section 6 introduces the metaheuristic learning algorithm that produces additional improvement by dramatically enhancing the quality of solutions obtained. Finally, Section 7 summarizes our findings.

2 Mathematical Formulation for Optimal Controlled Tabular Adjustment

The objective of synthetic tabular data is to closely mimic the original data, subject to obscuring sensitive cell values to a sufficient extent. By setting sensitive values to minimally safe values and constraining adjustments both locally (individual cells) and globally (overall measure of distortion), controlled tabular adjustment (CTA) is aimed at replacing original data by data that are comparable from a data analysis perspective. Cox and Danderkar (2004) provide additional approaches to preserving data accuracy and ease of use. Extensions that address statistical data analytic issues directly are presented in Cox and Kelly (2004) and Cox et al. (2004).

The underlying concept of CTA is simple: The value of each sensitive cell is replaced by an *adjusted value* selected to be at a safe distance from the original value. Danderkar and Cox (2002) suggest minimal adjustment, viz., to either the sensitive cell's lower or upper protection limit. Some or all nonsensitive cell values are then adjusted from their true values by small amounts to restore additivity to totals within the tabular system.

Within this framework, adjustments to nonsensitive cell values can be controlled in various ways. Selected nonsensitive cells, e.g., certain zero cells or totals, can be exempt from change through imposition of capacity constraints. Capacities are also used to confine nonsensitive adjustments to within meaningful limits such as sampling variability or total measurement error. One of several linear objective functions can be used to measure and assure minimum deviation. Some approaches, not discussed here, do not strictly adhere to the use of minimally safe values, replacing sensitive values by "safer" values (Cox and

Kelly 2004) or by "less safe" values based on enhanced protective capabilities of CTA (Cox and Danderkar 2004).

Tabular data systems with marginal entries can be represented by their system of linear equations in matrix form: $\boldsymbol{Ax} = \mathbf{0}$. Column vector $\boldsymbol{x}$ represents the tabulation cells of the system; $\boldsymbol{x}^*$ represents the original data. Matrix $\boldsymbol{A}$ is the *aggregation matrix* representing the tabular structure among the cells. The entries of $\boldsymbol{A}$ are –1, 0 or +1; each row of $\boldsymbol{A}$ corresponds to one *aggregation* (tabular equation) in which "+1" denotes a contributing internal cell and "–1" a total (*marginal*) cell. The mathematical structure of optimal synthetic tabular data is specified below by a mixed integer linear programming (MILP) formulation, containing binary and continuous variables, analogous to that introduced in Cox (2000).

Notation: $i = 1,\ldots, p$: denotes the p sensitive cells; $i = p+1,\ldots, n$: denotes the n-p nonsensitive cells; b_i = a binary (zero/one) variable denoting selection of the lower/upper limit for sensitive cell $i = 1,..., p$; l_i = lower deviation required to protect sensitive cell $i = 1,\ldots,p$; u_i = upper deviation required to protect sensitive cell $i = 1,\ldots,p$; y_i^+ = a nonnegative continuous variable identifying a "positive adjustment" for the gap to cell value i; y_i^- = a nonnegative continuous variable identifying a "negative adjustment" for the gap to cell value i; UB_i, LB_i = upper/lower cell bounds on change to cell i; c_i = cost per unit change in cell i.

MILP for Optimal Controlled Tabular Adjustment

$$Min \sum_{i=1}^{n} c_i (y_i^+ + y_i^-) \tag{1}$$

Subject to:

$$\boldsymbol{A}(\boldsymbol{y}^+ - \boldsymbol{y}^-) = \mathbf{0} \tag{2}$$

For $i = 1,...,n$:

$$0 \le y_i^+ \le UB_i \tag{3}$$

$$0 \le y_i^- \le LB_i \tag{4}$$

For $i = 1,..., p$:

$$y_i^+ = u_i b_i \qquad (b_i \text{ binary}) \tag{5}$$

$$y_i^- = l_i (1 - b_i) \tag{6}$$

After solving the MILP, the synthetic tabular data $\boldsymbol{t} = (t_i)$ is: $t_i = x_i^* + y_i^+ - y_i^-$. Equation (1) is the objective function, which minimizes the cost due to cell deviations. Two linear cost functions are commonly used, usually defined over deviation variables $y_i^+ + y_i^-$. The first involves coefficients $c_i = 1$, corresponding to minimizing the distortion measure "total absolute adjustment," and the other $c_i = 1/x_i^*$, corresponding to minimizing total percent absolute adjustment. CTA perturbs the sensitive cells until they are safe, i.e., until sensitive cell values are sufficiently far from their original values. Unless changes are carefully coordinated, this can create inconsistency in the tabular system, by causing the sums to no longer balance. Equation (2) maintains tabular consistency. Equations (3) and (4) are used to constrain the non-sensitive cell deviations. Usually, the upper bounds are computed using the estimated measurement errors for non-sensitive cells. Equations (5) and (6) ensure that the sensitive cells are set at their safe values. This is achieved by setting these cells at either their lower or upper protection limits. The protection limits for the cell include the minimum amount that must be added or subtracted from the true value to make the sensitive cells "safe". It can be noted that CTA offers increased immunity to disclosure attack because in CTA the sensitive cells are not highlighted and are replaced with a value. More importantly, sensitive cells are set at either their lower or upper limits. The intruder has no idea about the direction of perturbation. (Cox 1980; 2001)

It is possible that the CTA model gives rise to an infeasible problem if the number of sensitive cells in a particular row or column is large. The sensitive cell constraints in the model can be relaxed in the following manner to virtually eliminate these types of problems. Such a solution is acceptable since the constraints do not violate the important confidentiality protection condition.

$$y_i^+ \geq u_i b_i \tag{7}$$

$$y_i^- \geq l_i (1 - b_i) \tag{8}$$

Consider the following example, which illustrates how the mathematical programming formulation can be used to protect the sensitive cells in a 2-dimensional table as shown in Table 1. Cells (3, 1), (1, 2), and (3, 2) shown in bold have been identified as sensitive cells and the associated protection limits are shown in brackets. The upper and lower bounds for the non-sensitive cells are set at 20% of the original cell value. Table 2 shows the tabular data after solving the mathematical program. Cells with * indicate that they have been adjusted.

Table 1 Tabular Data before CTA

74	**17[0,37]**	85	176
71	51	30	152
1[0,21]	**9[0,29]**	36	46
146	77	151	374

Table 2 Tabular Data after CTA

75*	0*	85	160*
71	51	30	152
0*	29*	36	65*
146	80*	151	377*

The corresponding mathematical programming formulation is:

$$\textit{Minimize}: \sum_{i=1}^{4}\sum_{j=1}^{4}(y_{ij}^{+} + y_{ij}^{-})$$

subject to:

$$\sum_{j=1}^{3}(y_{ij}^{+} - y_{ij}^{-}) - y_{i4}^{+} + y_{i4}^{-} = 0 \qquad \textit{for each row } i = 1,\ldots,4$$

$$\sum_{i=1}^{3}(y_{ij}^{+} - y_{ij}^{-}) - y_{4j}^{+} + y_{4j}^{-} = 0 \qquad \textit{for each column } j = 1,\ldots,4$$

$$\begin{array}{ll}
y_{31}^{+} = 20 b_{31} & y_{31}^{-} = 1\,(1-b_{31}); \\
y_{12}^{+} = 20 b_{12} & y_{12}^{-} = 17\,(1-b_{12}) \\
y_{32}^{+} = 20 b_{32} & y_{32}^{-} = 9\,(1-b_{32}) \\
0 \le y_{11}^{+}, y_{11}^{-} \le 15 & 0 \le y_{21}^{+}, y_{21}^{-} \le 14 \\
0 \le y_{41}^{+}, y_{41}^{-} \le 29 & 0 \le y_{22}^{+}, y_{22}^{-} \le 10 \\
0 \le y_{42}^{+}, y_{42}^{-} \le 15 & 0 \le y_{13}^{+}, y_{13}^{-} \le 17 \\
0 \le y_{23}^{+}, y_{23}^{-} \le 6 & 0 \le y_{33}^{+}, y_{33}^{-} \le 7 \\
0 \le y_{43}^{+}, y_{43}^{-} \le 30 & 0 \le y_{14}^{+}, y_{14}^{-} \le 35 \\
0 \le y_{24}^{+}, y_{24}^{-} \le 30 & 0 \le y_{34}^{+}, y_{34}^{-} \le 25 \\
0 \le y_{44}^{+}, y_{44}^{-} \le 75 & b_{ij} \in [0,1]
\end{array}$$

The foregoing MILP formulation unfortunately can not be solved to optimality except for very small problems. As we show, the CPLEX solver for MILP problems requires an excessive amount of time to solve a problem of size no larger than 10x10x20.

3 Earlier Proposed Heuristics and Preliminary Numerical Testing

We first examine the simple heuristic methods proposed in the literature for the CTA problem and evaluate their effectiveness relative to a set of 2- and 3-dimensional test tables. The test problems include tables with varying attributes. To carry out the evaluation, forty-four 2-dimensional and six 3-dimensional tables are randomly generated using the following specifications:

- 2-dimensional tables range in size from 4x4 to 25x25. 3-dimensional tables have sizes: 5x5x2, 5x5x3, 5x5x4, 5x5x5, 5x5x6, and 5x5x7. The dimensions include only internal entries (not sums).
- Data values for internal tabular entries range from 0 to 1000 and are selected from a uniform distribution.
- 10% of the internal entries are selected randomly (uniformly distributed) and are assigned a value of 0. This is done to generate tables that closely resemble the real-life economic and social tabular data.
- For the 2-dimensional tables, two sets of tables are generated. The first set has 10% of the internal entries defined as sensitive. The second set has 30% of the internal entries defined as sensitive. The sensitive cells are distributed randomly (uniform) throughout the table. Marginal or sum cells are not defined as sensitive.

- For the 3-dimensional tables, 30% of the internal entries are defined as sensitive. The sensitive cells are distributed randomly (uniform) throughout the table. Marginal cells are not defined as sensitive.
- Sensitive entries must be assigned a value 20% greater than the original value or 20% smaller than the original value. All nonsensitive cells can be modified to values within 20% of their original values.

For 2-dimensional tables, the coefficient matrix is unimodular when the sensitive entries are integer, and consequently nonsensitive entries are automatically assigned to integer values. Solutions for 3-dimensional and other tables can produce fractional entries in nonsensitive cells. Integer values are often preferred for cosmetic purposes, and a typical and simple way to deal with this is to round fractional internal entries to their nearest integers and recompute totals.

For each method tested, two objective functions are evaluated. The first measure (*Unweighted*) minimizes the sum of the absolute changes. The second measure (*Weighted*) minimizes a relative measure that weights the absolute changes by the factor $1/x_i^*$. The methods are summarized below:

- *The ILOG-CPLEX 8.1 Optimizing (Exact) Solver*
- *Random Heuristic*: Sensitive entries are set to either their low value or high value with 0.5 probability. The nonsensitive entries are computed using a linear programming formulation. The simulation is run 100 times and the results are analyzed for worst, mean (average), and best case performance. In practice, the *Best Random* case is selected.
- *Ordered Heuristic*: Sensitive entries are ordered from smallest to largest value. Adjusted sensitive data values are assigned by alternating between the low value and the high value of the sensitive cell while moving through the ordered list. The one exception is when a cell value equals one or more of its corresponding totals, in which case both are assigned the same direction. The nonsensitive entries are computed using a linear programming formulation to evaluate the nonsensitive cells.

To evaluate the performance of the random and ordered heuristics, the results are compared to the optimal solutions found using the branch and bound procedure of CPLEX. For unweighted cases, the objective function is computed as the sum of the absolute perturbations, whereas weighted results are based on an objective function that normalizes the summands by $(1/x_i^*)$. Percent error equals: 100% (heuristic objective – optimal objective)/optimal objective.

Figure 1 displays the results obtained for the unweighted case with 2-dimensional tables that contain 10% sensitive entries. The figure shows results for tables ranging in size from 4x4 to 25x25. There is a single curve for the ordered heuristic and three curves for the random heuristic. The curves for the random heuristic provide mean, worst, and best solutions found during the 100 simulations.

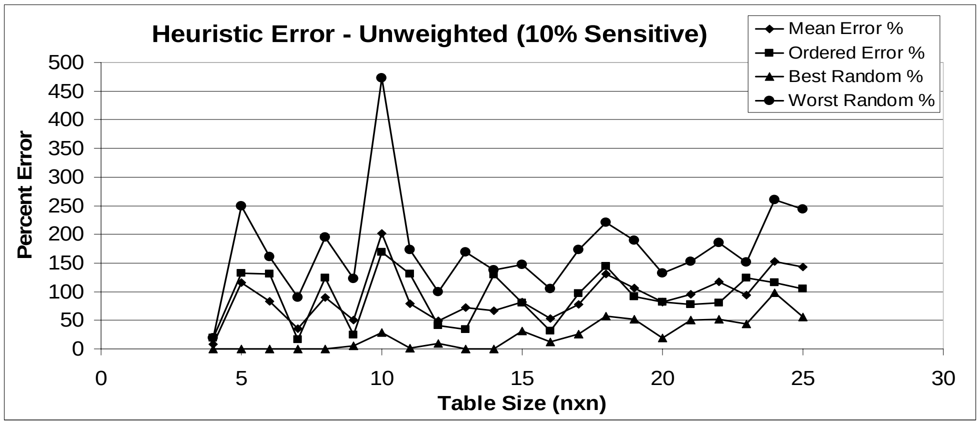

Fig. 1 Comparison of heuristics on 2-dimensional tables based on percent error

Figure 1 shows that best random performed best among the heuristics, but produces solutions that are far from optimal. It is interesting to note that the ordered heuristic method produces solutions of similar quality to the mean random result. Finally, it appears that error increases slightly with table size.

Figure 2 compares the approaches using the average relative change equal to the average of the absolute values obtained from (original value – new value)/original value.

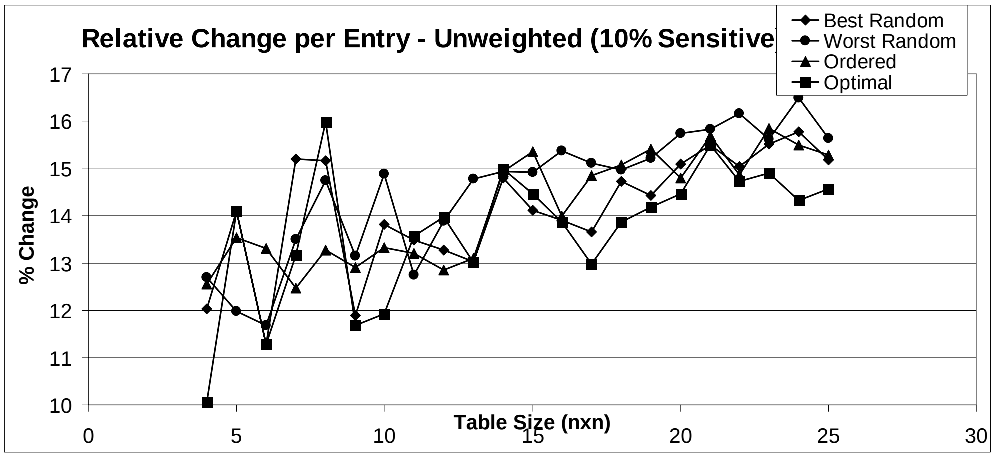

Fig. 2 Comparison of heuristics on 2-dimensional tables based on average percent change

Figure 2 shows that when considering relative changes, the best algorithm using an unweighted objective function is not clearly defined. Of course, a weighted objective function may provide more definitive results.

Next, 2-dimensional tables with 30% sensitive entries were processed. Except for the errors being larger, the results are analogous to those found for the tables with 10% sensitive entries. The relative changes for the 30% sensitive cell tables are also very similar to those found for the 10% sensitive cell tables, and thus are not included.

Figure 3 shows results for 3-dimensional tables ranging in size from 5x5x2 to 5x5x7 and containing 30% sensitive entries. The results that are obtained using an unweighted objective function are very similar to the results obtained for 2-dimensional tables.

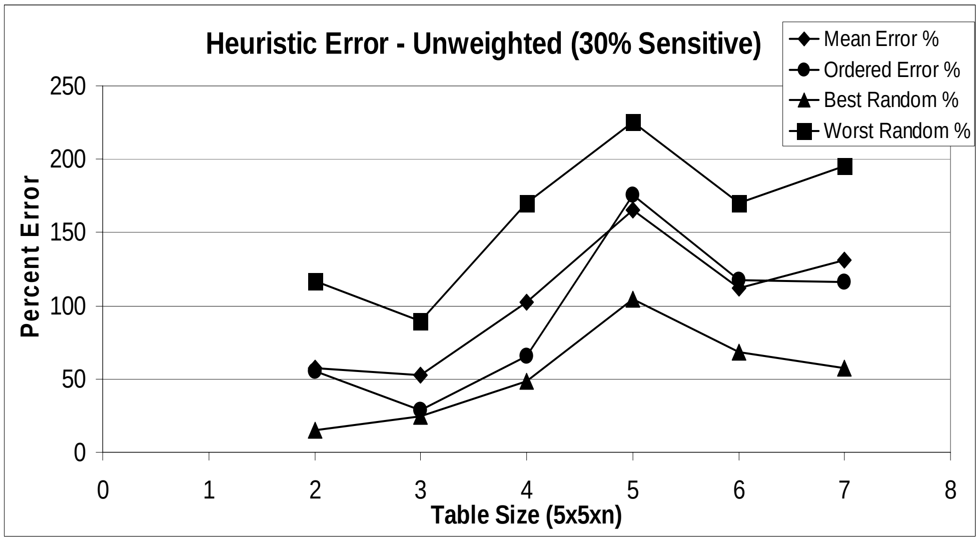

Fig. 3 Comparison of heuristics on 3-dimensional tables based on average percent error

The relative changes per entry for the 30% cases were found to be analogous to the results obtained for the 2-dimensional tables.

As discussed earlier, the linear programming solution for 3- and higher dimensional problems can produce solutions that contain fractional values. Figure 4 shows the percentage of tables for which the linear program produced computed fractional solutions when executing the random heuristic (100 runs). Clearly, the number of fractional values increases with table size. The random heuristic only produced fractional values for approximately 10% of the solutions generated. The best solution from 90% non-fractional solutions is reported. As mentioned earlier, when integer values are preferred in order to create cosmetically appealing entries for tables, then additional adjustment is appropriate, which fortunately has not been found difficult to perform.

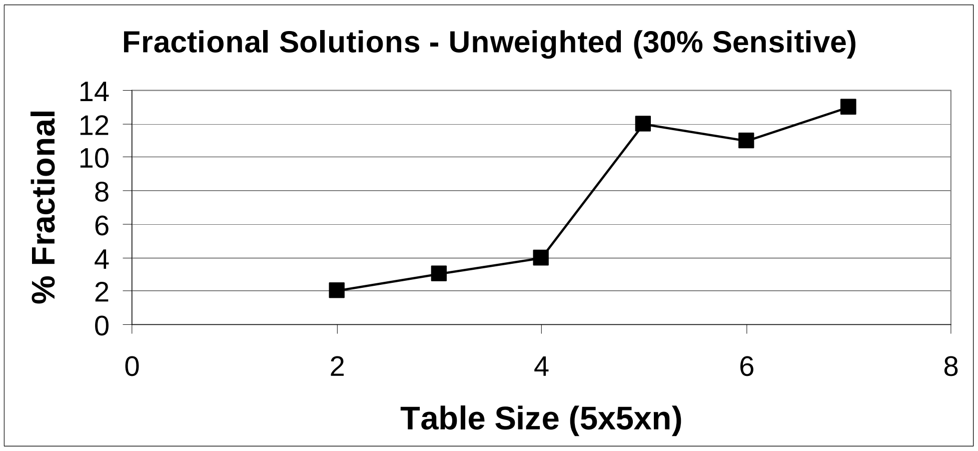

Fig. 4 Fractional solutions obtained for 3-dimensional tables using Random heuristic

To assess the impact of the type of objective function used, the previous analyses are also done using weighted objective functions. Figure 5 indicates that the best random heuristic is superior to the ordered heuristic, but still produces significant errors.

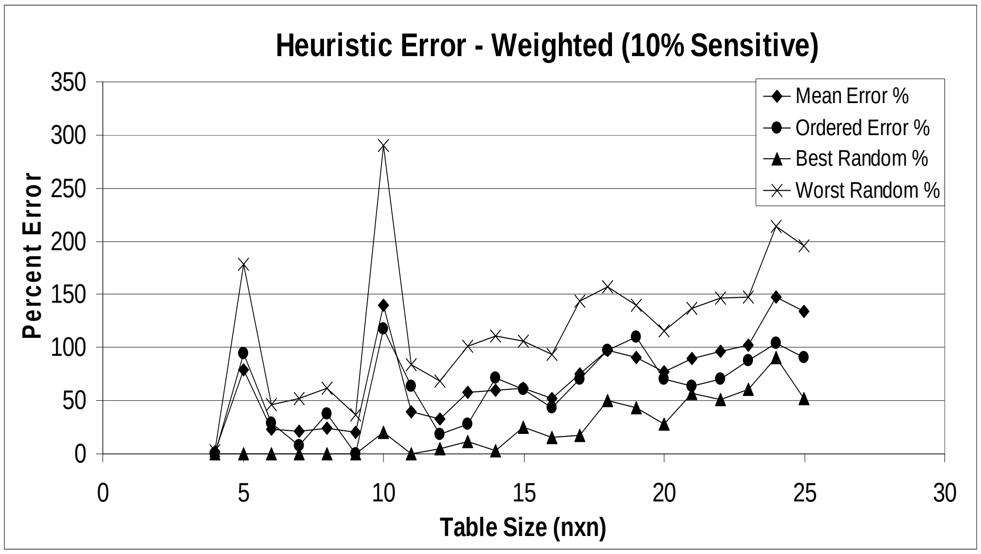

Fig. 5 Comparison of heuristics based on percent error using a weighted objective function

Figure 6 indicates all solution techniques produce similar quality solutions as the size of the table increases for tables with a larger percentage of sensitive cells when relative change per entry is used for comparison. This implies that the heuristics are a good option for such tables if the objective is to minimize the relative change per entry.

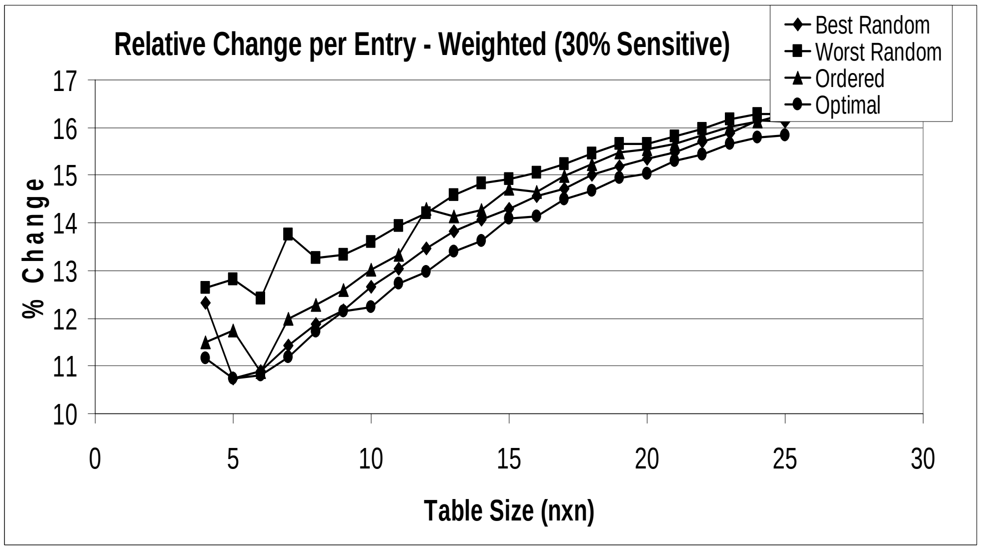

Fig. 6 Comparison of heuristics based on percent change using a weighted objective function

Results for 3-dimensional tables using weighted objective functions are consistent with the 2-dimensional results. In particular, weighted objective function causes larger perturbations but reduces relative perturbations. The number of fractional solutions is reduced when using the relative objective function. In fact, only the largest table (5x5x7) exhibited a 9% fractional solution, whereas the smaller tables did not produce any fractional solutions. It appears that the relative objective has an advantage in this regard.

The results of evaluating previously proposed heuristics for controlled tabular adjustment support the following observations:

- A weighted objective function that considers relative perturbation produces solutions with small average relative perturbation, but can potentially produce large absolute perturbations;
- The weighted objective function reduces the occurrence of fractional solutions when applied to 3-dimensional tables;
- The best random solution obtained over 100 random executions was shown to be superior to the best solution from the ordered heuristic in most cases;
- When objective function values were considered, the performance of all of the heuristics was poor, having errors in excess of 50%;
- As would be expected, for these problems of significantly limited size, the CPLEX solver produces the best solution regardless of the objective function used.

These results confirm that the exact solution approach works better than the heuristic approaches for small problems, but unfortunately CPLEX cannot be used to solve large problems, due to consuming excessive amounts of computation time. We hypothesize that the earlier heuristic methods evaluated here suffer because finding a feasible set of binary variables may be very hard. In particular, the heuristics may fail to generate a feasible solution for problems that have large numbers of sensitive cells. To combat this situation, we propose a new heuristic method that produces better results by combining the mathematical programming approach with the principles embodied in the Danderkar and Cox (2002) heuristics.

4 Stratified-Ordered Heuristic

The principle that underlies the two heuristics tested in the previous section is that in a good solution to the CTA problem approximately half of the sensitive cells will be set to their high values and the remainder will be set to their low values. This tends to reduce distortions to nonsensitive cells as balanced upper and lower adjustments to sensitive cells cancel adverse effects on the totals. Also, ordering the sensitive cells and alternatively setting them to their minimally low or high protection values tends to produce a solution whose grand total, and hence the overall mean, remains nearly unchanged. In this section, we endeavor to embed this ordering principle within the mathematical model previously described in Section 2.

Because computational requirements for our MILP roughly double with the addition of each binary variable, a sensible approach towards a computationally efficient, near-optimal algorithm is to group the sensitive cells and assign a unique binary variable to the entire group, with the result that all cells in a group are adjusted in the same direction. We first tried random grouping, which performed poorly. We then experimented with the idea of using a *stratified ordered heuristic* (or *s-ordered heuristic*) which orders sensitive cells from largest to smallest, and creates the groups by "skipping" through the ordering. This ensures greater group-to-group homogeneity so that large cells are less likely to be adjusted predominantly in the same direction. This produces an improvement in the optimum value of the objective function. As before, the exception is when a sensitive cell value equals one of its totals, in which case only a single cell is assigned to the group.

More precisely, let $m \geq 2$ be the number of groups. We add the following constraints to the original mathematical program.

For i=1 to m: $b_i = b_{i+m} = b_{i+2m} = \ldots b_{i+km}$ where $(i+km) \leq p$

The addition of these constraints to the original mathematical model reduces the number of binary variables to m. If $m = p$ then the solution is optimal, and if $m < p$ then the solution may or may not be optimal. However, for $m \leq 20$, the mathematical program can be solved in a reasonable amount of computer time. This set of constraints generated by s-ordered heuristic combines the power of the mathematical program with logical principles embodied in the heuristics.

Furthermore, the mathematical program can be enhanced with additional constraints (Cox and Kelly 2004; Cox et al. 2004) to improve the statistical characteristics of the solution. We apply the s-ordered heuristic with this model. In particular, we use groups of size, m, m-1, m-2, … to produce a range of results from which to choose a superior solution. The s-ordered heuristic overcomes a weakness of the ordered heuristic by not predefining the direction of change for each group. Whereas the ordered heuristic only evaluates one possible set of assignments, the s-ordered heuristic implicitly evaluates 2^m possible assignments. The assignment refers to the allocation of up and down directions to the sensitive cells by fixing the binary variables.

To determine the effectiveness of the s-ordered heuristic, sets of 2- and 3-dimensional test tables are randomly generated using the following specifications:

- 2-dimensional tables ranging in size from 4x4 to 25x25;
- 3-dimensional tables having sizes: nxnxn for *n* = 5,6,…,11,12…20;
- 3-dimensional tables having sizes: 10x10xn for *n* = 3,4,…,19,20;
- Data values for internal tabular entries range from 0 to 1000 and are selected from a uniform distribution;
- 10% of the internal entries are selected randomly (uniformly distributed) and are assigned a value of 0;
- For all tables, 30% of the internal entries are defined as sensitive. The sensitive cells are distributed randomly (uniform) throughout the table. Marginal cells are not defined as sensitive.
- Sensitive entries must be assigned a value 20% greater than the original value or 20% smaller than the original value. All nonsensitive cells can be modified to values within 20% of their original values.
- In all tables, an objective that minimizes the sum of absolute changes (unweighted) is used.

Figure 7 shows the performance of the heuristics compared to the optimal solution for moderately sized 2-dimensional tables. The optimal solution curve is not displayed because its information is embodied in the report of the percent error of heuristic solutions with respect to optimal. The random-100 and random-1000 results are obtained using one hundred random assignments and one thousand random assignments, respectively. The s-ordered heuristic (16) results are obtained using the s-ordered heuristic with m=16, which was chosen to provide solutions in approximately the same time as required by random-1000. The results indicate that the s-ordered heuristic is superior.

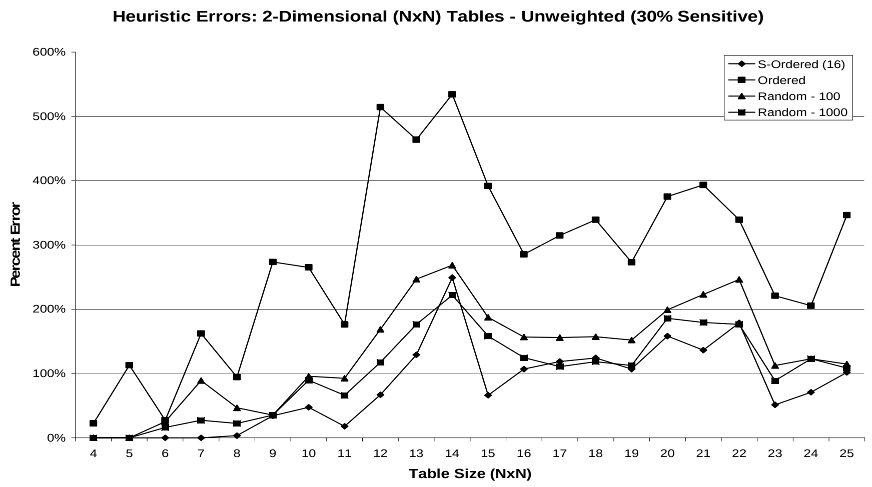

Fig. 7 S-ordered heuristic performance on 2-dimensional tables based on percent error

Figure 8 shows results for 3-dimensional tables. In these cases, optimal solutions could not be obtained for the larger tables. Thus, the results are compared to the best heuristic solution, which, in almost every case, is achieved by the s-ordered heuristic.

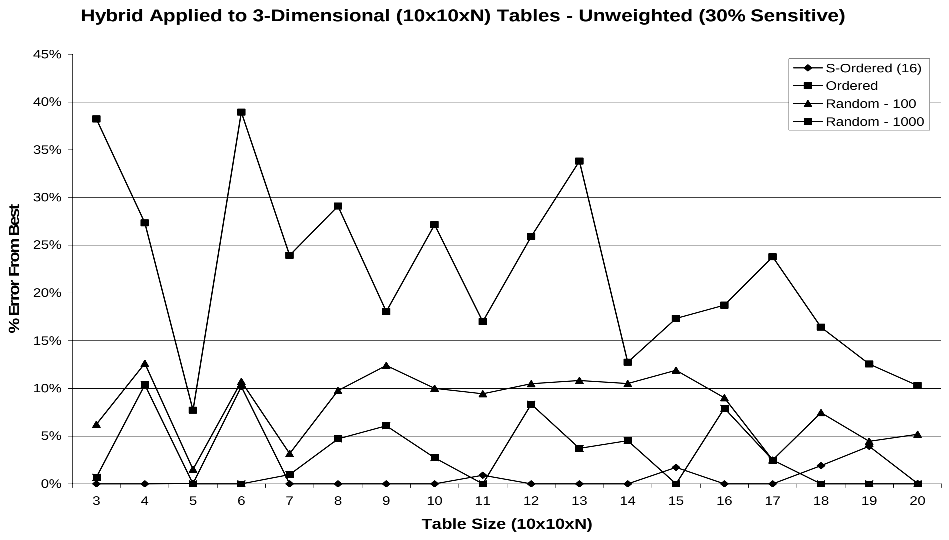

Fig. 8 S-ordered heuristic performance on 3-dimensional tables based on percent error

These results indicate that creating groupings of sensitive cells can significantly extend the applicability of the integer programming model. By using an ordering defined by cell value, reasonable solutions are produced.

In a final experiment with the s-ordered heuristic, we explored an advanced approach for building groups of cells. The principle here is to minimize the number of potential conflicts within each group so that assignments do not produce large perturbations to totals entries. First, the cells are grouped into m groups using the previous approach. For each group, we calculate the number of totals that are in common with each pair of cells. The number of internal totals interactions within a group is the *group score*. We then *swap* cells between groups to decrease the grand total of all group scores. Swaps are continued until no further score reduction is possible. The resulting groups are then used to populate the mixed integer program. This procedure is referred to as the s-ordered heuristic-with-swaps. Figure 9 (at the end of Section 5) demonstrates that this process improves the solutions approximately 10% on average.

5 Scatter Search for Enhancing the S-Ordered Heuristic

Using the mixed integer programming based approach becomes impractical when the number of tabular entries exceeds a thousand. For example, the 10x10x20 table reported in Fig 8 required 76 minutes of computational time on 2.8GHz, Pentium 4, 512 MB machine to process. To overcome this limitation, we implemented an evolutionary *scatter search* procedure to find solutions in reasonable time (Laguna and Marti 2003). Scatter search is designed to operate on a set of points, called *reference points*, which constitute good solutions obtained from previous efforts. The basis for defining "good" includes special criteria - typically related to diversity - that go beyond the objective function value. New points are then systematically generated combinations of the reference points. These combinations are generalized forms of linear combinations, accompanied by processes to adaptively enforce constraint-feasibility.

The set of points is considered diverse if its elements are "significantly" different from one another. We use Euclidean distances to determine how "close" a potential new point is from those in the reference set, in order to decide whether the point is included or discarded. The number of new solutions created depends on the quality of the solutions being combined. Specifically, when the best two reference solutions are combined, we generate up to five new solutions from their combinations, while when the worst two are combined we generate only one new solution.

In the process of searching for a global optimum, the combination method may not be able to generate solutions of sufficiently high quality to become members of the reference set. If the reference set does not change and all the combinations of solutions have been explored, a diversification step is triggered. This step consists of rebuilding the reference set to create a balance between solution quality and diversity. To preserve quality, a small set of the best (elite) solutions in the current reference set is used to seed the new reference set. Then, the diversification method is used to repopulate the reference set with solutions that are diverse with respect to the elite set. This reference set is used as the starting point for a new round of combinations. This method guarantees that a very good solution is found quickly.

We used the OptQuest solver to implement our scatter search method for the CTA problem. OptQuest uses a mixture of techniques including scatter search and advanced tabu search to find the right combination of decision variables to achieve the best possible results. During the search process, it also uses adaptive and neural network procedures to learn from past optimizations so that better solutions are obtained in a lesser amount of time (Laguna and Marti 2003).

Figure 9 shows the results of the scatter search method used in combination with the s-ordered heuristic-with-swaps. The proposed heuristic performed very well on all instances compared to the ordered and random heuristics. Figure 9 also provides results from taking the best solution obtained from OptQuest-2000 using $m = 9,10,...,16$ (which encompasses the cases $m = 1,...,8$). Hence, it is also referred to as OptQuest-2000-swap9-16. This experiment provided the best solutions in all cases, except that it doubled the computation time required to run the problem for m=16. It should be noted that for all tables for N≤ 10 the scatter search heuristic solutions were shown to be optimal. In the case of tables for larger N, CPLEX could not be used to determine the possible optimality of the scatter search solutions due to its inordinate solution time requirements. So, the results are compared to the best heuristic solution, which, in almost every case, is achieved by the s-ordered heuristic.

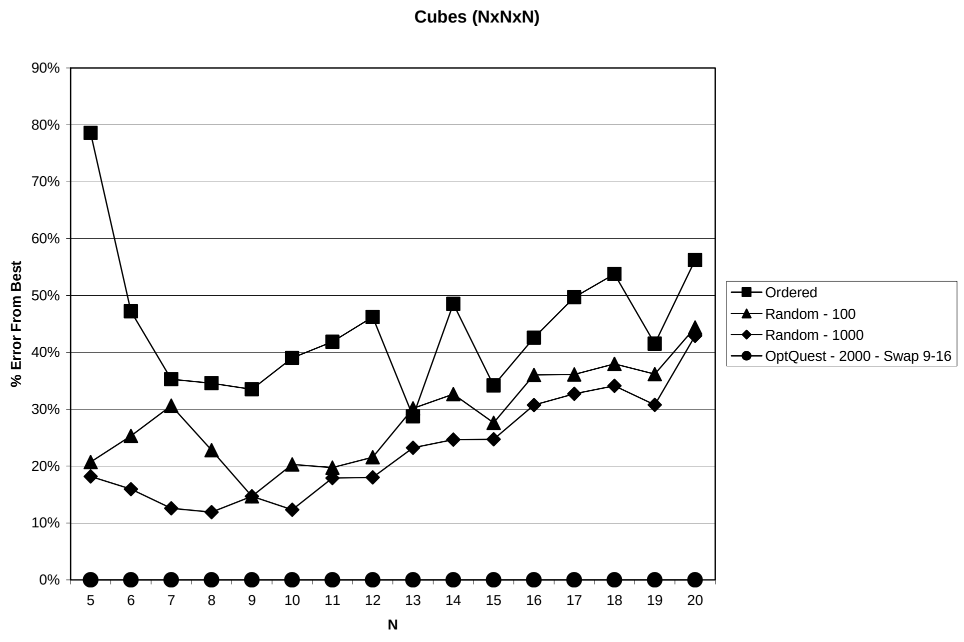

Fig. 9 Performance of scatter search combined with the s-ordered heuristic-with-swaps on cubic 3-dimensional tables

6 Meta-heuristic Learning Algorithm for Forecasting Directions for Binary Variables

6.1 Learning Algorithm

The grouping heuristics proposed in the previous section significantly reduced the problem size and thereby quickly solved the resulting integer program. However, these methods nevertheless failed to produce satisfactory solutions for problems beyond a relatively limited size. The best heuristic solution was at least 50% inferior to the optimal solution for all moderately large 2-dimensional problems. Moreover, the heuristics exhibited considerable variation in the solution quality produced (see Figure 10). These experiments demonstrate the importance of reducing the size of the integer programs for gaining computational efficiency. We attribute the inferior performance of these methods on larger problems to their inability to predict and set appropriate values for a subset of variables. In this section, we show that a metaheuristic learning strategy for fixing a subset of variables to appropriate values offers an opportunity to generate high-quality solutions without confronting the typical drawback of consuming vast amounts of computer time to discover such solutions.

6.2 Parametric Image Learning Metaheuristic

Our learning procedure is an adaptive memory learning metaheuristic that creates a strategic image of part of the problem to generate information about problem characteristics. Such processes have been used successfully in the fixed charge context (Glover et al. 2004), and are the basis for a class of adaptive memory metaheuristic procedures for mixed integer programming proposed in Glover (2006). Adapted to the present setting, the basic idea is to introduce parameters that penalize a variable's violation of integer feasibility, and to drive selected subsets of variables in preferred directions, e.g., toward 0 or 1, under metaheuristic guidance.

In the CTA problem, we are interested in identifying appropriate directions for selected subsets of binary variables, which are then tentatively fixed at their preferred values. The resulting reduced problem is then solved much more readily than the original problem, providing an iterative process that results in high-quality (optimal or near optimal) solutions while expending only a small fraction of the computational effort required by a more traditional integer programming solution approach. We utilize this strategy to develop a parametric objective function approach to generate information about behavior of binary variables in the following manner.

We represent the objective function in the compact form: Minimize $x_0 = cx$, where x is a set of binary variables used to protect sensitive cells. We refer to the "1" direction as UP and the "0" direction as DN in our framework. These are called *goal conditions* (denoted as $x_j^{'}$) because we do not seek to enforce UP and DN directions by imposing them as constraints in the manner of customary branch and bound method, but rather indirectly by incorporating them into the objective function of the linear programming relaxation. Let N^+ and N^- denote selected subsets of N whose union is denoted by $N^{'}$ and whose elements contain UP and DN goal conditions, respectively. Let $x^{'}$ denote the associated goal imposed solution vector and let M denote a very large positive number used to impose the goal conditions.

$$(LP^{'})\,\text{Minimize } x_0^{'} = \sum_{j\in N^-}\left(c_j + M\right)x_j + \sum_{j\in N^+}\left(c_j - M\right)x_j + \sum_{j\in N/(N^+ + N^-)} c_j x_j \qquad (9)$$

Problem $(LP^{'})$ targets imposed down and up goal conditions by using incentive mechanism driven by the penalty M. Binary variables included in subset N^- are induced to go in the DN direction and binary variable in subset N^+ are induced to go in UP direction. Remaining variables are free to select their own favorable directions or to receive values between 0 and 1. Thus, in short, we are solving a continuous linear programming problem with penalty coefficients in the objective as a way to gain insight about good values to assign to the binary variables.

6.3 Goal Infeasibility and Resistance

If a variable indeed favors a particular direction, then it will achieve its targeted goal. Otherwise, we say that it demonstrates *resistance* to its imposed goal. We say that an optimal LP solution $x = x''$ is goal infeasible if one of the following two "violations" occurs:

For some $j \in N^{+}, x''_j < x'_j$ (V-UP)

For some $j \in N^{-}, x''_j > x'_j$ (V-DN)

We call a variable x_j associated with violation (V-UP) or (V-DN) a *goal infeasible* variable, and we create a measure called an *overt resistance* (β UP, β DN), based on goal conditions, to learn about variable predilection for a particular direction as follows.

For (V-UP), $\beta UP_j = x'_j - x''_j$ (10)

For (V-DN) $\beta DN_j = x''_j - x'_j$ (11)

An absence of a goal violation means that zero overt resistance occurs. Sometimes it is possible that a variable may resist its goal condition even though it does not violate its goal condition. We can compute this effect by making use of reduced costs in the following manner. We call this resistance a *potential resistance* ($\delta UP, \delta DN$).

$$\delta UP_j = M + c_j + RC_j \tag{12}$$
$$\delta DN_j = -(-M + c_j + RC_j) \tag{13}$$

where RC_j is the reduced cost for variable x_j.

The trial solution vector may also contain variables that have not yet been assigned penalties, or that have had previously assigned penalties removed. We use their solution values in the problem (LP) to create *free resistances* ($\alpha UP, \alpha DN$) in the following manner.

$$\alpha UP_j = 1 - x_j \tag{14}$$
$$\alpha DN_j = x_j \tag{15}$$

6.4 Experimental Design to Exploit the Parametric Image

We incorporate experimental design in our approach as a foundation for its learning component. First, the parametric image of objective function is generated using a goal vector. A diversified sample of goal vectors is generated and resistance measures are recorded to estimate directional effects. We do not employ random sampling, as used in network design problems (Karger 1999), because it is not efficient in terms of the number of tests required to estimate a preferred value for each variable.

Experimental design methods are often used to identify significant factors controlling a performance measure (for example, to determine machine speed and pressure levels that will produce a product of desired quality) (Montgomery 1984). These methods can also be used to estimate main effects and interaction effects of binary variables using a smaller number of unbiased samples than random sampling (Lewis 2004).

The main effect of a binary variable can be defined as the mean value that the variable achieves on a performance measure. We incorporate the three types of resistance measures previously indicated in conjunction with the objective function value to provide measures for identifying preferred directions for the problem variables. A test run in our experimental design process is composed of creating and solving a continuous LP problem that takes (9) as its objective. These test runs are inexpensive in the sense that they can be implemented using efficient linear programming post optimizations rather than by restarting the LP solution from scratch.

Despite the computational efficiency of individual runs, the number of test runs grows exponentially with the number of variables used for a full factorial design, and we seek a better alternative. The basic purpose of full factorial design is to estimate interaction effects as well as main effects. However, we hypothesize that interaction effects have a negligible influence compared to main effects, due to the sparsity of the variable effects in the integer programming context. Thus, we focus on estimating main effects without concern for interaction effects in the present setting.

Fractional factorial design can be used to target desired variable effects using confounding techniques. This design has an ability to reduce the number of trial vectors considerably. For example, a problem with 10 binary variables will need 1024 test runs with full factorial design, but will need only 16 test runs to estimate main effects with fractional factorial design. If necessary, we can also add more runs to a fractional factorial design to improve the accuracy of the method, employing the strategy known as sequential experimentation.

One possible method to implement a fractional factorial design is to generate a set of goal vectors over a complete set of binary variables. This method has the disadvantage that it would eliminate the use of free resistances as defined by (14) and (15). Instead, we prefer to partition binary variables into groups and run different experimental design runs over these subsets while keeping variables in other subsets free, thereby generating information on free resistances. This also makes it possible to analyze the problem from different angles by conducting different experimental design runs. Test runs provide information on goal resistances and objective values. Experimental design then identifies the main effect of each variable for each performance measure.

Recently, Lewis (2004) has used experimental design techniques in integer programming, making use of elastic constraints to avoid problem infeasibility. However, our method differs from this approach at a fundamental level. Instead of resorting to elastic constraints, our parametric image approach avoids infeasibility in a way that allows us to focus on studying the behavior of variables in different circumstances to learn about their optimal directions. Second, in contrast to Lewis (2004), who used the objective function as the sole performance measure, we make use of measures based on different goal resistances in addition to the objective function.

Our approach of selecting different performance measures for finding true main effects is motivated by the fact that information about the desirability of different choices is captured in different forms by different rules. This information can be used more effectively by means of a strategy that combines the rules in aggregation rather than by using a strategy of selecting different rules at different times (Glover and Laguna 1997). The learning algorithm used to fix directions for a specified subset of variables can be summarized as below. The details for carrying out these steps are elaborated subsequently.

6.5 Parametric Image Learning Algorithm

```
1. Group p binary variables into lastK subsets of size n such that
   lastK= |_p / n_|
2. Construct goal vectors for parametric image process using fractional
   factorial design. (Refer to Appendix for further details.)
3. Set an upper bound on the objective function to induce trial solutions to
   come from better regions.
4. Run fractional factorial experimental design as:
   For subsets K = 1… lastK
       For test runs T = 1…lastT
           Construct the parametric image of the objective function
           using a partial goal vector.
           Solve the resulting linear programming relaxation.
           Compute overt, potential, free resistances and objective
           function value.
           Record these performance measures into pertinent
           performance recording vectors [PV].
       End T
   End K
5. Relative to each variable, compute the main effect of measures except
   free resistance measures as:
   For performance attribute A = 1…lastA (except free resistance)
       For variables P=1…lastP
           Compute main effect ME [A][P] of variable 'p' in attribute
           'a' as :
             {
                 For Experiment K = 1… lastK
                     For test runs T = 1…lastT
                         If ($x'_p$ = DN) then
                             ME[A][P] = ME[A][P] -
                 PV[A[K][T]
                         If ($x'_p$ = UP) then
                             ME[A][P] = ME[A][P] +
                 PV[A][K][T]
                     End T
                 End K
             }
       End P
   End A
6. Relative to each variable, compute main effect of the free resistance
   measures as:
       For variables P=1…lastP
```

```
                Compute main effect ME[A][P] as
                  {
                        For experiment K = 1…lastK
                              For test runs T= 1…lastT
                                    If ( $x_p$ < 0.5 ) then
                                          ME[A][P] = ME[A][P] + 1
                                    If ( $x_p$ > 0.5 ) then
                                          ME[A][P] = ME[A][P]  - 1

                              End T
                        End K
                  }
          End P
```

7. Compute final score for each variable using persistent voting principle as:

```
   For variables P=1…lastP
         Final Score [P] = 0
               For A = 1…lastA (includes free resistance measure)
                     If  (ME [A][P] > 0) then
                           Final Score [P] = Final Score [P] + 1
                     If  (ME [a][p] < 0) then
                           Final Score [P] = Final Score [P] – 1
               End A
   End P
```

8. Rank variables P = 1…lastP in descending order of the absolute values of final scores.
9. Set cutoff ‘c’ to fix direction for the variables.
10. Fix directions for binary variables as:

```
         For variables P=1…c
               If (Final Score [P] > 0) then
                     $x_p$ = 0
               If (Final Score [P] < 0) then
                     $x_p$ = 1
         End P
```

11. Solve the resulting mixed integer programming problem.

6.6 Discussions and Elaboration of the Method

Variables are grouped into K subsets in step 1 in a random manner. The rationale for using a random assignment is to avoid generating an interaction effect. By contrast, a process of grouping variables from a particular row or column together can produce significant interaction effects because of tabular additivity. The fractional factorial design we employ confounds interaction effects with the aim of reducing the number of test runs. Selecting sensitive cells in a random fashion encourages them to exhibit minor interaction effects because of weak tabular connectivity.

The logic of our earlier ordered heuristic, which assigns up and down directions for ordered cells in an alternating fashion, is consistent with this finding. Thus, the ordered heuristic tries to capitalize on a positive two-factor interaction effect. The heuristic embodies a subtle limitation, however, which serves to undermine its efficacy. Sometimes, the heuristic may assign either plus or minus directions to all cells in a row or a column, thereby increasing the absolute adjustment. For example, consider a 4x4 table in which cells (1,1), (2,4), (3,1), and (4,4) are sensitive with protection limits of 40,35,30,25 respectively. The ordered heuristic would cause very large adjustments to nonsensitive cells in this case. This might be a primary reason why the ordered heuristic did not perform well in our experiments. We overcome this deficiency in our present approach by exploiting the following design.

A parametric image of the objective function is generated as follows. A typical test run contains target directions for a subset of variables and free directions for remaining variables. We can use these targeted directions to generate a parametric image of the objective function as:

$$x_o' = \sum_{j\in N^-}\left(c_j + M\right)x_j + \sum_{j\in N^+}\left(c_j - M\right)x_j + \sum_{j\in N/(N^+ + N^-)} c_j\, x_j \tag{16}$$

The new model has the effect of inducing variables with a DN goal condition to receive a value of 0 and variables with an UP goal condition to receive a value of 1, while allowing remaining variables to take arbitrary values (without being induced to move in a particular direction).

Steps 5 and 6 compute the average effect of a variable for a given measure. This basically computes the average change in the performance measure when a binary variable is changed from 0 to 1. This method is widely used in experimental design to compute average effects (Montgomery 1984) because it groups observations into two sets and then checks on average whether there is any difference in performance between the two sets. For example, if the DN direction sum is higher than the UP direction sum, this signals a negative effect, implying that a performance measure would decrease if a binary variable were set to 1 instead of 0.

We record performance measures in 3-dimensional vectors for each variable, where row dimension refers to the performance measure, column dimension refers to the experiment and page dimension refers to the test run from the experiment. As shown in steps 5 and 6, to calculate the main effect for a measure, we sum over performance values computed with respect to all test runs from all experiments for that measure. We subsequently record the main effects of variables in a 2-dimensional vector in which row dimension refers to the performance measure and column dimension refers to a binary variable.

We rank binary variables in descending order according to the absolute values of their final scores and select a subset of these variables to receive fixed directions. The cutoff level was decided using experimental evaluation. We found 45% and 70% as cutoff levels for “small” and “big” tables respectively, in the sense that these levels generated high-quality solutions in a reasonable amount of time. If the size of the table is below (above) 15x15, then it is referred to as a small (large) table, respectively. The results section shows in detail how the percentage of fixed variables affected solution quality and time. Use of 0 as threshold in the final step is mainly conceptual in nature, as it means that variables with positive final scores prefer the DN direction and those with negative final scores prefer the UP direction. The chosen cutoff level ensures that variables chosen to be fixed will be those that have sufficiently high absolute final scores, thereby offering adequate support for the chosen directions.

6.7 Performance of the Learning Algorithm for 2-Dimensional Tables

We implemented the learning algorithm using C++, ILOG-Concert Technology 1.2, and ILOG-CPLEX 8.1. Figure 10 shows the performance of our proposed method compared to other variable fixing heuristics. It was extremely time-consuming to run larger problem instances to optimality using the version of default CPLEX. For example, we ran the 25x25 problem using the default CPLEX method on a 2GB RAM and 3.2GHz workstation for 24 hours. Unfortunately, the best solution found by CPLEX (after 19 hours and 35 minutes) still exhibited an optimality gap of 9.6%.

Consequently, we needed a computationally efficient alternative to compute a better lower bound, which is essential for measuring the optimality gap. Cox et al. (2005) proposed a set partitioning relaxation for generating a tighter lower bound on the objective in the CTA context. We used the lower bound as a proxy for an optimum value for computing the optimality gap for larger instances. Lower bounds were reliable in the sense that they were consistently very close to the optimal values for those problems where an optimal solution could be verified (by running CPLEX for a period of time that does not exceed practical feasibility). In particular, for these problems involving 2-dimensional tables, restricted in size to no more than 18 rows and columns to permit them to be solved by CPLEX, the optimality gap was verified to be approximately 1%. For example, for the 18x18 problem, the computed lower bound was 9736 compared to the optimum value of 9850, representing a gap of 1.15%. In Figure 10, the "Learning Method (optimal)" curve identifies the optimality gap with respect to the known optimal value, and the "Learning Method (lower bound)" curve identifies the optimality gap with respect to the lower bound.

We found our learning method to yield significant improvements in reducing the optimality gap across the entire 2-dimensional test set, as demonstrated by Figure 10. Optimality gap values obtained by the methods described in preceding sections degraded considerably for the larger problem instances. For example, using these earlier methods, the mean gap for the 25x25 table was 117.6% compared to the overall mean gap of 70% for smaller problems. In either case, the results were disappointing. By contrast, the learning method performed dramatically better, consistently generating high-quality solutions irrespective of the problem size, giving an overall mean gap of 6% and a total gap of 5.72% for the 25x25 problem.

We define *prediction accuracy* to be the percentage of variables that are correctly assigned their optimal values, from a selected set of the "top" (highest scoring) variables identified. The prediction accuracy of our method, for a 14x14 problem which contains 69 variables, was 85.5% for the top 10% of the problem variables (6 correct decisions out of 7 fixed variables). In order to analyze the tradeoff between solution quality and time, we fixed only the top 15% of the variables for the 17x17 problem. We found a better solution (objective value = 9206) than our reported solution (objective value = 9460), although it was at the expense of computational efficiency. For this particular experiment, the result of fixing fewer variables caused the number of nodes processed to increase from 2400 to 72600 and the solution time to increase from 16.38 sec to 520 sec. We believe this increase in the computation time does not warrant reducing the number of fixed variables in order to achieve a modest gain in solution quality.

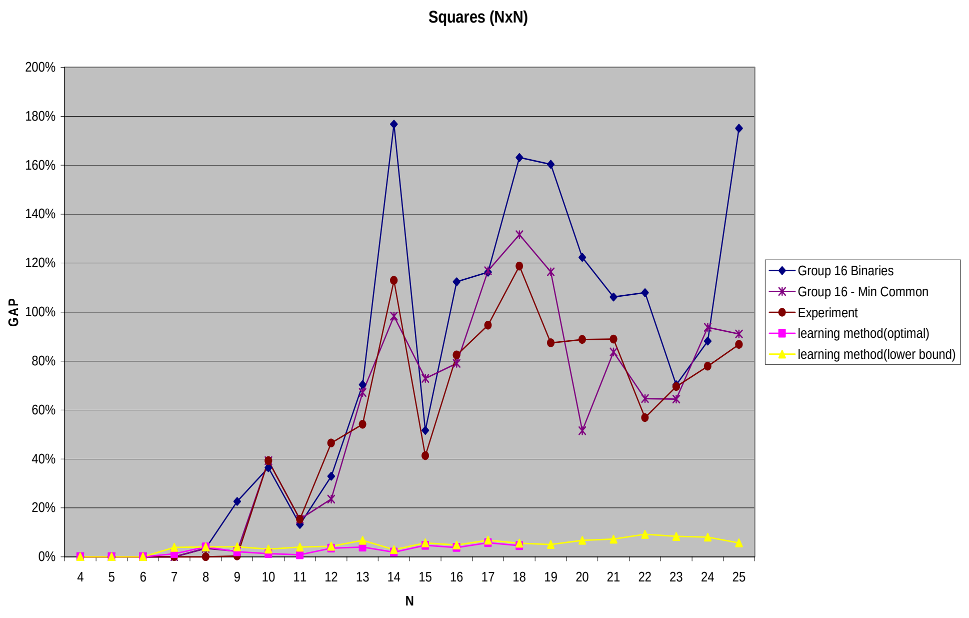

Fig. 10 Performance of proposed learning method on optimality gaps

7 Conclusions and Remarks

This study has undertaken an extensive set of comparative computations tests and analyses to evaluate the relative performance of alternative methods for the controlled tabular adjustment (CTA) model. Our preliminary tests compared previously proposed heuristics to the exact CPLEX method. The outcomes showed that the exact procedure yields solutions superior to those of earlier heuristic approaches, but is unable to solve problems of modest size within a reasonable amount of time.

To overcome these limitations of previous approaches, we have introduced a stratified (s-ordered) heuristic that combines the exact mathematical programming approach with constructive heuristics suggested in Danderkar and Cox (2002). Numeric simulations indicate that the s-ordered technique has the ability to produce better solutions than the previous heuristics in reasonable time, and has the added advantage of being able to find reasonable solutions to highly constrained problems, but is limited to problems that remain of modest dimension. We then showed that using an evolutionary scatter search approach in place of the exact CPLEX solver yields improved results and makes it possible to handle problems of much greater size, though the approach still is unable to overcome the combinatorial complexity of these problems to achieve solutions that appear attractive in relation to optimality bounds.

Finally, we demonstrate that a special metaheuristic learning method based on parametric image processes leads to significant additional improvements by generating solutions of greatly improved quality. In particular, the learning method succeeds in reducing the optimality gap for the problems tested from an overall average of 70% to an average of 6%. The true distance from a theoretical optimum is likely to be somewhat smaller still, since the gap is based on an imprecise bound.

We anticipate that opportunities exist to improve our results further. Interactions between binary variables are likely to be present, especially variables corresponding to cells sharing the same tabular equation. Our approach does not explicitly incorporate interactions into the factorial design portion of the learning procedure, because to do this would drastically multiply the number of trial solutions to be compared and tested. Instead, we address such interactions at another level through the solution of the partial integer program. Evidence of the ability to accommodate such interactions is provided by the significant improvements produced by our method compared to the heuristic and exact methods, including the extended variants in which these methods incorporate aggregated variables designed to capture interaction effects.

Nevertheless, it seems likely that there may be benefits in examining interactions from additional perspectives in future research, by undertaking to account for dependencies in advance of solving the parametric LP problems. An appealing strategy comes from Cox et al. (2005), involving the solution of set partitioning sub-problems as a foundation for creating special types of aggregated variables. Extensions of the learning approach that rely more fully on ideas of parametric tabu search are also relevant to explore.

Appendix

Full factorial design is used to estimate main effects as well as interaction effects. This design uses all possible combinations of the levels of factors to estimate these effects. For a binary program, these test runs constitute all possible leaf nodes of a binary tree. The tree traversal methods can be used to compute these vectors. Another alternative is to use bit-wise operators to generate full factorial design. The number of test runs in this design grows exponentially with the binary variables. Fractional factorial design builds an experimental design on a smaller full factorial design of the chosen subset of binary variables, thereby requiring a smaller number of test runs. Our learning algorithm uses fractional factorial design to generate goal vectors. The method can be described as follows. See Montgomery (1984) for further details.

1. Compute the minimum number of variables (r) needed to generate a basic design for the fractional factorial experiment (FFE) as $r = \lfloor \log_2^n \rfloor + 1$

2. Compute the minimum number of test runs (t) needed for conducting FFE as $t = 2^r$ test runs

3. Compute the number of generators required to generate goals for remaining variables as m = n-r

4. Generate a full factorial basic design for first r binary variables ($x_1 \ldots x_r$).

5. Use 'm' 2,3,…,h factorial interactions to compute values for remaining variables as.

```
For M = r +1…LastM
    For T= 1…LastT
        Code 0 as –1 and code 1 as 1 for binary variables values used in
        interaction.
        Compute the product of coded values of interacting variables for
        test t
        If (product > 0) then
            x[m][t] = 1
            If (product  < 0) then
                    x[m][t] = 0
        End T
End M
```

The following example, which contains six binary variables, illustrates the forgoing method.

1. Minimum number of binary variables needed for basic design: r = 2+ 1=3
2. Minimum number of test required: $t = 2^3 = 8$
3. Number of generators required: m = 6-3 =3
4. The resolution III design is feasible because 2 factor and 3 factor interactions suffice to complete FFE.
5. A full factorial basic design for first r binary variables is

Table 3 Full factorial design for 3 binary variables

Test Run	x_1	x_2	x_3
1	0	0	0
2	1	0	0
3	0	1	0
4	1	1	0
5	0	0	1
6	1	0	1
7	0	1	1
8	1	1	1

6. Construct the FFE table using 2-factor and 3-factor interactions. We choose $x_1\ x_2\ x_3$, $x_1\ x_2$, and $x_1\ x_3$ interaction effects to generate values for x_4, x_5, and x_6. For example, for test run 1 the value of x_4 is 0 because the value of coded product of $x_1\ x_2\ x_3$ is 0. The complete FFE design is shown in Table 4.

Table 4 Fractional Factorial design table for binary a program with 6 binary variables.

Test Run	x_1	x_2	x_3	$x_4 = x_1\ x_2\ x_3$	$x_5 = x_1\ x_2$	$x_6 = x_1\ x_3$
1	0	0	0	0	1	1
2	1	0	0	1	0	0
3	0	1	0	1	0	1
4	1	1	0	0	1	0
5	0	0	1	1	1	0
6	1	0	1	0	0	1
7	0	1	1	0	0	0
8	1	1	1	1	1	1

References (Bibliography)

Cox, L.H. (1980). Suppression Methodology and Statistical Disclosure Control. *Journal of the American Statistical Association,* 75, 377-385.

Cox, L.H. (1981). Linear Sensitivity Measures in Statistical Disclosure Control. *Journal of Statistical Planning and Inference,* 5, 153-164.

Cox, L.H. (2000). Discussion (on Session 49: Statistical Disclosure Control for Establishment Data). In ICES II: The Second International Conference on Establishment Surveys-Survey methods for businesses, farms and institutions, Invited Papers, Alexandria, VA: American Statistical Association, 904-907.

Cox, L.H. (2001). Chapter 8: Disclosure Risk for Tabular Economic Data. In Doyle, P., Lane, J.I., Theeuwes, J.J.M. and Zayatz, L.V. (eds.), Confidentiality, Disclosure and Data Access: Theory and practical applications for statistical agencies, Amsterdam: North-Holland, 167-184.

Cox, L.H., and Danderkar, R.A. (2004). A Disclosure Limitation Method for Tabular Data That Preserves Accuracy and Ease-of-Use. In proceedings of the 2002 FCSM Statistical Policy Conference, Washington, DC: Office of Management and Budget, 15-30.

Cox, L.H., and Kelly, J. P. (2004). Balancing Data Quality and Confidentiality for Tabular Data. Proceedings of the UNECE/EUROSTAT Work Session on Statistical Data Confidentiality, Luxembourg, 7-9 April, 2003, *Monographs of Official Statistics*, Luxembourg: Eurostat., 2004, 11-23.

Cox, L.H., Kelly, J.P., and Patil, R.J. (2004). Preserving Quality and Confidentiality for Multivariate Tabular Data. Proceedings of Privacy in Statistical Databases 2004 (PSD 2004), Barcelona, 9-11 June, 2004, *Lecture Notes in Computer Science, 3050*, New York: Springer Verlag, 87-98.

Cox, L.H., Kelly, J.P., and Patil, R.J. (2005). Computational Aspects of Controlled Tabular Adjustment: Algorithm and Analysis. The Next Wave in Computer, Optimization and Decision Technologies (B. Golden, S. Raghavan and E. Wasil, eds.), Boston: Kluwer, 45-59.

Danderkar, R.A., and Cox, L. H. (2002). Synthetic Tabular Data-An Alternative to Complementary Cell Suppression. Manuscript.

Fischetti, M., and Salazar, J.J. (1999). Models and Algorithms for the 2-Dimensional Cell Suppression Problem in Statistical Disclosure Control. *Mathematical Programming,* 84, 283-312.

Fischetti, M., and Salazar, J.J. (2000). Solving the Cell Suppression Problem on Tabular Data with Linear Constraints. *Management Science,* 47, 1008-1026.

Glover, F. (1977). Heuristics for Integer Programming Using Surrogate Constraints. *Decision Sciences,* 8, 156-166.

Glover, F. (2006). Parametric Tabu Search Methods for Mixed Integer Programming. *Computers and Operations Research,* 33(9), 2449-2494.

Glover, F. Amini, M., and Kochenberger, G. (2004). Parametric Ghost Image Processes for Fixed-Charge Problems: A Study of Transportation Networks. *Journal of Heuristics*, 11(4), 307-336.

Glover F., and Laguna M. (1997). Tabu Search. Kluwer Academic Publishers, Boston.

Karger, D.R. (1999). Random Sampling in Cut, Flow, and Network Design Problems. *Mathematics of Operations Research,* 24(2), 383-413.

Kelly, J., Golden, B., and Assad, A. (1992). Cell Suppression: Disclosure Protection for Sensitive Tabular Data. *Networks,* 22, 397-417.

Laguna, M., and Marti, R. (2003). Scatter Search: Methodology and Implementations in C, Kluwer Academic Publishers, Boston.

Lewis, M. W. (2004). Solving Fixed Charge Multi-Commodity Network Design Problems using Guided Design Search. University of Mississippi, Hearin Center Technical Report , HCES-01-04.

Montgomery, D.C. (1984). Design and Analysis of Experiments. John Wiley and Sons, New York, NY.

Willenborg, L., and de Waal, T. D. (1996). Statistical Disclosure Control in Practice. *Lecture Notes in Statistics,* 111, Springer, New York.

Willenborg, L., and de Waal, T. D. (2001). Elements of Statistical Disclosure Control. *Lecture Notes in Statistics,* 155, Springer, New York.