European Journal of Operational Research 250 (2016) 91–100

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Obtaining cell counts for contingency tables from rounded conditional frequencies

Andrew J. Sage [a], Stephen E. Wright [b,*]

[a] Department of Statistics, Iowa State University, Ames, Iowa, United States
[b] Department of Statistics, Miami University, Oxford, Ohio, United States

ARTICLE INFO

Article history:
Received 21 January 2014
Accepted 7 September 2015
Available online 12 September 2015

Keywords:
OR in government
Integer linear programming
Statistical disclosure control
Tabular data
Fast Fourier transform

ABSTRACT

We present an integer linear programming formulation and solution procedure for determining the tightest bounds on cell counts in a multi-way contingency table, given knowledge of a corresponding derived two-way table of rounded conditional probabilities and the sample size. The problem has application in statistical disclosure limitation, which is concerned with releasing useful data to the public and researchers while also preserving privacy and confidentiality. Previous work on this problem invoked the simplifying assumption that the conditionals were released as fractions in lowest terms, rather than the more realistic and complicated setting of rounded decimal values that is treated here. The proposed procedure finds all possible counts for each cell and runs fast enough to handle moderately sized tables.

© 2015 Elsevier B.V. and Association of European Operational Research Societies (EURO) within the International Federation of Operational Research Societies (IFORS). All rights reserved.

1. Introduction

Statistical disclosure control is concerned with privacy guarantees when releasing data that might otherwise identify, or reveal information about, specific individuals or organizations. Releasing data in summarized form greatly reduces disclosure risk, but does not eliminate the risk altogether. In particular, a contingency table of frequency counts potentially reveals information if the table includes any cells having small or zero counts. On the other hand, a table is less useful for statistical inference if it overly aggregates information. This trade-off generally leads to the release of a modified or aggregated version of the table that carries somewhat less information than the original. In deciding which particular form to release, a crucial step is to determine the tightest possible bounds that can be inferred about the individual cell counts in the original table. Such bounds can be used to assess both the disclosure risk and the statistical utility of the released information. The present paper addresses the open question of how to calculate these bounds when the contingency table is released in the form of a two-way table of rounded conditional frequencies.

A *two-way contingency table* is a two-dimensional array of whole numbers, and the row and column sums of such an array are referred to as *marginal* counts. The *conditional row probabilities* are given by dividing each entry in the array by the marginal sum for its row. As an example, the contingency tables shown in Tables 1(a) and 1(b) have the same *sample size* (sum of all entries) and also lead to the same conditional row probabilities, which are shown in Table 1(c). Wright and Smucker (2014) showed that Tables 1(a) and 1(b) are the only contingency tables of sample size 48 having the row conditionals shown in Table 1(c). Because Tables 1(a) and 1(b) have the same second row, knowledge of Table 1(c) and the sample size therefore exposes the actual counts in the second row. The question is whether those counts could be obscured somewhat by converting the exact fractions in Table 1(c) into rounded decimal expressions. In Section 3 we show that two-digit rounding does obscure the counts in this example whereas three-digit rounding does not.

Various approaches are used to limit the disclosure risk of contingency tables, such as choosing a more heavily aggregated summary (e.g., marginal counts only), suppressing some cells altogether, or perturbing cell counts slightly (Hundepool, Domingo-Ferrer, Franconi, Giessing, Nordholt, Spicer, & de Wolf, 2012). These often rely on operations research techniques to limit risk and evaluate risk-utility trade-offs (Almeida & Carvalho, 2005; Castro, 2006; 2011; 2012; Cox, 1995; Cox & Ernst, 1982; Fischetti & Salazar-González, 1999; Hernández-García & Salazar-González, 2014; Kelly, Golden, & Assad, 1990; Kelly, Golden, Assad, & Baker, 1990; Muralidhar & Sarathy, 2006; Salazar-González, 2004; 2005; 2006; 2008). Over the past decade researchers have also explored the possibility of releasing observed conditional probabilities, which retain some statistical utility insofar as odds and ratios of odds are preserved (Slavković, 2010). To date, the work on disclosure risk for tables of conditionals has focused primarily on two-way contingency tables, such as two-way

* Corresponding author. Tel.: +15135291837.
E-mail address: wrightse@MiamiOH.edu (S.E. Wright).

http://dx.doi.org/10.1016/j.ejor.2015.09.011
0377-2217/© 2015 Elsevier B.V. and Association of European Operational Research Societies (EURO) within the International Federation of Operational Research Societies (IFORS). All rights reserved.

Table 1
Two contingency tables, (a) and (b), with sample size 48 and row conditionals (c).

(a)			(b)			(c)	
3	4	7	9	12	21	3/7	4/7
5	3	8	5	3	8	5/8	3/8
6	9	15	4	6	10	2/5	3/5
10	8	18	5	4	9	5/9	4/9
24	24	48	23	25	48		

rearrangements of multi-way tables in which some subset of the observed variables are treated as the conditions (i.e., predictors), some are treated as the responses, and perhaps others are omitted altogether (by aggregating over their values) (Fienberg & Slavković, 2005; Slavković, 2004; Smucker & Slavković, 2008; Smucker, Slavković, & Zhu, 2012).

Our problem description and methods are framed in terms of two-way contingency tables, but the same approach can also be used to obtain cell bounds on multi-way contingency tables that have been reshaped as two-way tables by designating some variables as predictors, other variables as responses, and perhaps omitting some variables altogether. We refer the reader to Smucker et al. (2012), Wright and Smucker (2013), Wright and Smucker (2014) for more information on how and why such reshaping might be performed.

Here is a concrete example of a multi-way table using public data that nevertheless provides a nice illustration of how several variables might be grouped into predictor, response, or omitted variables. We consider an 8-way table ($N = 48,842$) from the 1993 U.S. Current Population Survey (CPS), a monthly survey that collects demographic and other data of interest. Table 2 gives information about the qualitative variables measured in this study. Imagine that data such as these comprised, say, all the adults in a given small city. There might be some concern in releasing it in full detail. In particular, small cell counts (including zeros) could potentially be used to identify the salary range of specific individuals. Other variables in this data set might also be considered sensitive information for some people, depending on their personal circumstances. Among the eight variables here, we would almost always consider age, race and sex as natural choices for possible predictors, whereas the other variables (except salary) could be considered either a predictor or response depending on the question at hand. Likewise, any of the variables might conceivably be omitted (implying an aggregation of counts). Moreover, levels within a variable might be combined into a smaller number of levels, such as reducing the three age ranges (for variable X_1 of Table 2) to two age ranges. In any case, the value of row conditionals now becomes clearer: over all people in the study who satisfy a given set of demographic predictors, the row conditionals tell us what proportion have a given (say) education level and salary level. The actual counts could well be of less interest than the proportions to decision-makers and the general public. But privacy considerations and statistical inference would require recovering information about those underlying counts from the stated proportions.

Table 2
CPS variables and their levels.

	Variable	Levels
X_1	Age	< 25, 25–55, > 55
X_2	Employment	Gov't, private, self-employed, other
X_3	Education	$<$ HS, HS, college, bachelor, bachelor+
X_4	Marital status	Married, unmarried
X_5	Race	Non-white, white
X_6	Sex	Female, *M*ale
X_7	Hours worked	< 40, 40, > 40
X_8	Salary	< 50, 50+

The mathematical structure of the corresponding cell-bounding problem was recently elucidated for the simpler context in which conditional probabilities are presented as *unrounded* fractions Slavković, Zhu, and Petrović, Wright and Smucker (2013). Under that idealization, it was demonstrated that the (upper or lower) bounding problem for each cell can be reduced to an integer linear knapsack problem. Wright and Smucker (2014) subsequently showed that cell bounds and possible counts for an entire two-way table of unrounded conditionals can be obtained quickly with an algorithm that shares intermediate solution information among large groups of cells in the table. They used that capability to explore disclosure risk for various rearrangements of the data represented in Table 2 under the assumption that a data snooper somehow determined the true unrounded fractions for each conditional probability.

The present article examines the two-way cell-bounding problem in the more complicated and realistic setting of *rounded* conditionals. Several of the works cited above (especially Smucker et al., 2012, Slavković, Zhu, & Petrović) provide light commentary, heuristics or preliminary results on issues relating to cell-bounding from rounded conditionals. So far there have been no substantive attempts to provide a general procedure for identifying tightest cell bounds. Here we formulate the bounding problem for each cell as a pair of integer linear optimization problems and show how these can be decomposed into two types of simpler subproblems. One type of subproblem is solvable in closed form and the other can be addressed by adapting some of the ideas presented in Wright and Smucker (2014). The decomposition provides useful knowledge about the cell-bounding problem structure and identifies all possible cell *counts* rather than merely the cell bounds (which is the most that could be obtained with integer programming methods). We also provide simple examples showing that: (a) rounded conditionals can lead to considerably wider tightest bounds than the corresponding unrounded fractions, and (b) a single-digit change in the rounding precision sometimes means the difference between revealing many cell counts and revealing none.

A full discussion of how to evaluate statistical utility and disclosure risk goes beyond the scope of this paper, but the basic issues are as follows. On the one hand, disclosure risk is minimized by avoiding the revelation of small counts (magnitude 3 or smaller, say) in the contingency table, and this in turn suggests that very narrow cell bounds are undesirable. This is easily understood, so our discussion of the examples in this paper tends to focus on disclosure. On the other hand, statistical utility refers to the ability to perform estimation or hypothesis testing and obtain results only slightly weaker than what one could obtain by applying standard statistical methods to the actual table of counts. Intuitively, such utility is maximized by having a more accurate representation of the underlying table, but the picture is more complicated than that. Salazar-González (2004) (also Salazar-González, 2005) has presented a fine overview of statistical disclosure limitation and its trade-offs from an operations research perspective.

Our method finds bounds on row marginals along with information about individual cell counts, with the latter then implying bounds on the remaining marginals. Performing inference on the basis of such bounds is an area of ongoing research, and a few methods have been proposed for the purpose (see Slavković, Zhu, and Petrović, Dobra, 2012 and references cited therein). Given that the implied bounds on marginals could be quite wide when disclosure risk is avoided, we are led to ask how badly data utility might be compromised. Slavković and Lee (2010) have provided some encouraging results on this last point. They proposed a method for disclosure limitation based on the following idea: round the actual row conditional probabilities to nearby decimal-place approximations, rewrite those approximate conditionals as fractions themselves, and then randomly select a table of counts from the collection of contingency tables for which the new fractions are the conditional probabilities. Releasing

such a "synthetic" contingency table is similar to, but slightly different than, releasing rounded conditionals themselves. They examined the utility-risk trade-off between releasing such a table and the traditional approach of releasing a table that preserves marginals (rather than conditionals) by swapping variables in the underlying microdata; this latter approach was used with data tabulations from the 2000 U.S. Census. For the particular example they considered (see Section 3 of the current paper) they found that statistical inference performed directly on the released table was not compromised when the table preserved conditionals, whereas it was seriously affected when marginals were preserved instead. At present, there is still a lot of scope for determining efficient and effective methods for inference based on conditionals, and for evaluating risk versus utility.

In the next section, we define the mathematical problem, formulate it as an integer programming problem, and then describe a decomposition method for solving it. In Section 3, we give some examples illustrating features that distinguish the solutions obtained from rounded versus unrounded conditionals. The fourth section presents some computational results illustrating actual running time versus the theoretical value from Section 2, along with a preliminary comparison to the naive use of a commercial solver. The final section provides some additional discussion of the results within the larger context of statistical disclosure limitation.

2. The mathematical problem and its solution

Consider a two-way contingency table having I rows and J columns, for which the actual (observed) count in cell ij is denoted o_{ij}. The data provider plans to release only the sample size $N = \sum_{i,j} o_{ij}$ and the conditional probabilities $o_{ij}/\Sigma_k o_{ik}$, which give the relative proportions within each column across the row i. However, the conditionals will first be rounded or perturbed to nearby numbers $p_{j|i}$ expressible with a fixed number of decimal places: $|p_{j|i} - o_{ij}/\sum_k o_{ik}| \le \epsilon$ for all i, j. Assuming a value has been specified for ϵ (either provided with, or inferred from, the released table), the problem we address is to identify the tightest bounds for the nonnegative integers n_{ij} satisfying $\sum_{i,j} n_{ij} = N$ and $|p_{j|i} - n_{ij}/\sum_k n_{ik}| \le \epsilon$. In fact, it is often useful to know precisely which values each cell may take; the method described below can do that as well. With a small adjustment, we can also address the related situation in which the rounding or perturbation procedure guarantees a *strict* inequality of the form $|p_{j|i} - o_{ij}/\sum_k o_{ik}| < \epsilon$ instead of the weak inequality indicated above.

2.1. Formulation

To formulate the cell-bounding problem, we let N_i denote the possible row sum (i.e., denominator) for row i. For greater generality, we also allow for the possibility that prior bounds might be known for some row sums or cell counts. Our task is to solve the following collection of $2IJ$ integer linear optimization problems, one pair for each cell $\overline{ij}$:

$$\text{min/max } n_{\overline{ij}} \text{ over all integers } n_{ij}, N_i \text{ subject to} \tag{1}$$

$$\sum_i N_i = N, \tag{2}$$

$$L_i \le N_i \le U_i, \quad \text{for all } i, \tag{3}$$

$$\sum_j n_{ij} = N_i, \quad \text{for all } i, \tag{4}$$

$$(p_{j|i} - \epsilon)N_i \le n_{ij} \le (p_{j|i} + \epsilon)N_i, \quad \text{for all } i, j, \tag{5}$$

$$l_{ij} \le n_{ij} \le u_{ij}, \quad \text{for all } i, j. \tag{6}$$

The inequality (5) expresses the constraint on the rounding error in cell ij, whereas (3) and (6) are *a priori* bounds on row sums and cell counts. The bounds are assumed to be integer-valued with $U_i \ge L_i > 0$ and $u_{ij} \ge l_{ij} \ge 0$.

We now comment briefly on the computational complexity of these problems. Determining integer feasibility of (2)–(6) is NP-hard for arbitrarily chosen positive integers L_i, U_i, l_{ij}, u_{ij}, rational numbers $p_{j|i} > 0$ (with or without $1 = \sum_j p_{j|i}$) and rational $\epsilon \ge 0$. To see this, it suffices to show that (2)–(6) capture as a special case the following well-known NP-complete decision problem (Nemhauser & Wolsey, 1988): given positive integers $a_1, \ldots, a_I$ and b, determine if there exist values $x_1, \ldots, x_I \in \{0, 1\}$ satisfying $\sum_i a_i x_i = b$. For an instance of this knapsack-equality decision problem, we can define data for an instance of (2)–(6) by

$$J = 2, \quad N = b + \sum_i a_i, \quad L_i = a_i,$$

$$U_i = 2a_i, \quad \epsilon = 1/(4 \max_i\{a_i^2\})$$

$$l_{ij} = \begin{cases} 1, & \text{if } j = 1, \\ a_i - 1, & \text{if } j = 2, \end{cases} \quad u_{ij} = \begin{cases} 2, & \text{if } j = 1, \\ 2(a_i - 1), & \text{if } j = 2, \end{cases}$$

$$p_{j|i} = \begin{cases} 1/a_i, & \text{if } j = 1, \\ 1 - 1/a_i, & \text{if } j = 2. \end{cases}$$

The equivalence of the resulting instance of (2)–(6) with the standard decision problem above is then given by the correspondence $x_i = n_{i1} - 1$, where we also take $N_i = a_i n_{i1}$ and $n_{i2} = N_i - n_{i1}$. These last two equations must be satisfied by all integer solutions of (2)–(6) because of the particular choice of $\epsilon > 0$; in other words, we might as well take $\epsilon = 0$. Conversely, as noted previously (Smucker et al., 2012), Slavković, Zhu, and Petrović, the special case of *unrounded* conditionals (i.e., any instance with $\epsilon = 0$) admits a direct reformulation as a collection of equality-constrained knapsack problems.

To summarize, we expect the problems (1)–(6) to be a good deal more challenging than linear programming and at least as hard to solve in practice as knapsack problems. On the other hand, omitting the constraint (5) leaves a totally unimodular system that could be handled with linear programming or network optimization. In fact, it can be addressed in closed form as a special case of Theorem 2.1 below.

In the next two subsections we describe a procedure for solving these optimization problems.

2.2. Decomposition

We propose a solution framework based on the observation that each instance (1)–(6) can be decomposed into two families of subproblems:

- Given a value of N_i, what cell counts n_{ij} (if any) within row i yield N_i as a row sum?
- Given a set of N_i-values for each i, which of those N_i can be included in a sum equal to N?

The solution procedure begins by addressing the if-any aspect of the first subproblem to identify a set of N_i-values, which is then considered in the context of the second subproblem. For each N_i admitted by the second subproblem, we follow up by solving the first subproblem to obtain the actual cell counts. In other words, the framework is roughly analogous to eliminating all n_{ij} in favor of N_i, then solving a system that involves only the N_i-values, and finally substituting back to recover n_{ij}.

The key to this approach is that the first subproblem above admits closed-form solutions in both directions, namely, when restricting the choice of N_i during the elimination of n_{ij} and again when recovering n_{ij} from each such N_i. The former amounts to some easily derived inequalities relating N_i to n_{ij}, whereas the latter relies on the

ability to guarantee that those inequalities are tight and that n_{ij} can take all consecutive integer values within the bounds implied by a given choice of N_i. The rest of this subsection presents these closed forms, proves their validity, and indicates their formal role in the framework outlined above. Examples involving small contingency tables are sketched in Section 3.

The decomposition makes use of tight explicit bounds on n_{ij} imposed by N_i through the constraints (4)–(6). To calculate those bounds, we first rewrite the conditions (5 and 6) equivalently in the form

$$l_{ij}(N_i) \le n_{ij} \le u_{ij}(N_i), \quad \text{for all } i, j, \tag{7}$$

where the left and right sides are defined as

$$l_{ij}(N_i) := \max\{l_{ij}, \lceil (p_{j|i} - \epsilon)N_i \rceil\},$$
$$u_{ij}(N_i) := \min\{u_{ij}, \lfloor (p_{j|i} + \epsilon)N_i \rfloor\}.$$

Here we use the notation $\lfloor z \rfloor$ for the greatest integer that is no larger than z, and $\lceil z \rceil$ for the least integer that is no smaller than z. Taking into account the possible gap between the left and right sides of (4) when the n_{ij}-values are at their bounds (6), we define

$$n^-_{ij}(N_i) := \max\{l_{ij}(N_i),\ N_i - \textstyle\sum_{j' \ne j} u_{ij'}(N_i)\},$$
$$n^+_{ij}(N_i) := \min\{u_{ij}(N_i),\ N_i - \textstyle\sum_{j' \ne j} l_{ij'}(N_i)\}$$

and then consider the inequalities

$$n^-_{ij}(N_i) \le n_{ij} \le n^+_{ij}(N_i). \tag{8}$$

With this notation, the decomposition is described by the following result. It shows how to select row sums N_i for consideration and how to recover cell counts n_{ij} from N_i.

Theorem 2.1. *Consider a positive integer N_i for a fixed row i. There exist integers n_{ij} satisfying (4)–(6) if and only if N_i satisfies*

$$l_{ij}(N_i) \le u_{ij}(N_i), \quad \text{for all } j, \tag{9}$$

and

$$\textstyle\sum_j l_{ij}(N_i) \le N_i \le \sum_j u_{ij}(N_i). \tag{10}$$

Moreover, for each such N_i, the attainable integer values of n_{ij} in (4)–(6) are those satisfying (8).

Proof. First suppose there are integers n_{ij} satisfying (4)–(6), so that (7) and therefore (9) both hold. Combining (4) with the left inequality in (7) gives us $N_i = \sum_j n_{ij} \ge \sum_j l_{ij}(N_i)$, and hence we obtain the left inequality of (10). At the same time, (4) and the right inequality in (7) yield $n_{ij} = N_i - \sum_{j' \ne j} n_{ij'} \ge N_i - \sum_{j' \ne j} u_{ij'}$, so the left inequality of (8) holds. In a similar fashion, we can verify the right inequalities of (10) and (8). This proves the claimed necessity of (8)–(10).

Conversely, suppose that N_i satisfies (9) and (10). We need to express N_i as a sum of integers n_{ij} satisfying (7). Also, the final statement of the theorem requires showing that each integer value of n_{ij} indicated in (8) can be attained. Without loss of generality (by permuting column indices as needed), we demonstrate such attainment for the final column $j = J$. To this end, consider a value n_{ij} between n^-_{ij} and n^+_{ij}. Defining $m_k := \sum_{j \le k} u_{ij}(N_i) + \sum_{k < j < J} l_{ij}(N_i)$ for $k = 0, 1, \ldots, J-1$, we see that $m_{J-1} \ge N_i - n_{ij}$ and that m_k is nondecreasing in k. For the smallest k such that $m_k \ge N_i - n_{ij}$, it is readily verified that the choice

$$n_{ij} = \begin{cases} u_{ij}(N_i), & \text{if } 0 < j < k, \\ l_{ij}(N_i), & \text{if } k < j < J, \\ N_i - n_{ij} - m_k + u_{ik}(N_i), & \text{if } j = k > 0 \end{cases}$$

satisfies both (4) and (7). □

The decomposition presented in Theorem 2.1 validates the three-phase solution process outlined earlier, which can now be described more completely.

For each row i, the conditions (9 and 10) are easily checked for each value of $N_i \in \{1, \ldots, N\}$, as are the simple bounds (3). In this way we identify the collection, denoted $\mathcal{N}_i$, of row sums that accommodate the rounding constraints and bounds for all cells in row i. This eliminates the cell counts n_{ij} momentarily and constitutes the first phase of the solution procedure.

In the second phase, we extract those elements $N_i \in \mathcal{N}_i$ that can be combined with admissible sums from other rows to satisfy the remaining constraint (2). More precisely, we calculate the set $\mathcal{N}_i^{\text{feas}} := \mathcal{N}_i \cap \left(N - \sum_{i' \ne i} \mathcal{N}_{i'}\right)$, where the sum of sets is understood in the Minkowski sense (i.e., all numbers expressible as sums $\sum_{i' \ne i} \nu_{i'}$ with $\nu_{i'} \in \mathcal{N}_{i'}$). As described in Section 2.3, this calculation be accomplished quickly for fairly large two-way contingency tables.

The third and final phase of the solution procedure consists of substituting each $N_i \in \mathcal{N}_i^{\text{feas}}$ into the left and right sides of (8) to find the attainable values of n_{ij}. Clearly, the extremes sought by the objective in (1)–(6) lie among the values $n^-_{ij}(N_i)$ and $n^+_{ij}(N_i)$ for some choices of N_i. It should be noted that the functions $n^-_{ij}(\cdot)$ and $n^+_{ij}(\cdot)$ are not necessarily monotonic, so cell-specific optima might not correspond to extreme elements of $\mathcal{N}_i^{\text{feas}}$.

We also note that N_i and ϵ together determine how many values n_{ij} lie in the range (8). It can be shown that $\lfloor p_{j|i} N_i \rfloor \le n^-_{ij}(N_i) \le n^+_{ij}(N_i) \le \lceil p_{j|i} N_i \rceil$ whenever $N_i < 1/\epsilon$, so there are at most two integer values n_{ij} satisfying (8) when N_i is sufficiently small. For $N_i \ge 1/\epsilon$, the right and left sides in (8) differ by no more than $2\lceil \epsilon N_i \rceil$.

The first and third phases above can be carried out in closed form with $O(J)$ comparisons and arithmetic operations for each N_i considered, and therefore they require $O(IJN)$ such operations overall. As will be seen in the next subsection, the second phase requires at most $O(IN \log I \log N)$ and so that component tends to dominate the running time. Section 4 presents the results of some computational tests to see whether this time estimate might be realized in practice. Note that all calculations can be carried out using integer arithmetic, if desired.

We now indicate how to adapt this method to address strict rounding inequalities of the form $|p_{j|i} - o_{ij} / \sum_k o_{ik}| < \epsilon$, which are sometimes used in consistent rounding instead of the corresponding weak inequalities. As an example, for 3-digit rounding with $\epsilon = 0.001$ we might allow any number strictly between 0.134 and 0.135 to be rounded to either of those two values, but 0.1340 could not be converted to 0.135 (as would be allowed by the weak inequality). In the problem formulation, this clearly amounts to replacing the weak inequalities in (5) with strict inequalities. Although this potentially makes the integer programming representation more complicated, it is handled easily by the decomposition presented above. The only modification needed is to replace the definitions of $l_{ij}(N_i)$ and $u_{ij}(N_i)$ with these:

$$l_{ij}(N_i) := \max\{l_{ij}, \lfloor (p_{j|i} - \epsilon)N_i \rfloor + 1\},$$
$$u_{ij}(N_i) := \min\{u_{ij}, \lceil (p_{j|i} + \epsilon)N_i \rceil - 1\}.$$

All the remaining details in this section apply without further change.

2.3. Details on the second phase

We now look more closely at the second phase of the solution procedure presented in Section 2.2. Given sets $\mathcal{N}_1, \ldots, \mathcal{N}_I$ of positive integers, we must calculate $\mathcal{N}_i^{\text{feas}} = \mathcal{N}_i \cap \left(N - \sum_{i' \ne i} \mathcal{N}_{i'}\right)$ for each i. First we note that the calculation of $\mathcal{N}_i^{\text{feas}}$ requires only those elements of $\sum_{i' \ne i} \mathcal{N}_{i'}$ that do not exceed the sample size N. Hence, when adding two sets of the form $\mathcal{N}_{i'}$ it suffices to use a pair of nested loops over subsets of $\{0, \ldots, N\}$. Doing this for each row i and each $i' \ne i$ would amount to $O(I^2 N^2)$ operations overall, which is reasonable if I and N are not too large. When I or N is large (in the thousands, say), it is worthwhile to seek greater efficiency.

For each $N_i \in \mathcal{N}_i$ we need to determine whether $N - N_i$ can be expressed as a sum $\sum_{i' \neq i} N_{i'}$ for some choice of $N_{i'} \in \mathcal{N}_{i'}$. If each set $\mathcal{N}_i$ were a uniformly spaced set of the form $\{r_0 + \lambda r \mid \lambda \in [\lambda^-, \lambda^+] \cap \mathbb{Z}\}$, then this subproblem would constitute an equality-constrained, bounded knapsack problem. Indeed, the cell-bounding problem for *unrounded* conditionals leads precisely to such a situation, in which case each set $\mathcal{N}_i$ simply consists of consecutive multiples of a single number r_i (the lowest common denominator for row i). Wright and Smucker (2014) therefore proposed a refinement of a standard dynamic programming algorithm for knapsack problems. A dynamic programming perspective is particularly attractive in this context because we must address multiple values of N_i within each row i and we also hope to exploit redundancy in the calculations used for distinct rows.

Unfortunately, $\mathcal{N}_i$ in the present setting does not have such a nice form, but in general admits arbitrarily small or large integer gaps between its elements. We note, however, that one special case can be treated by a direct extension of the approach cited above: in the absence of *a priori* finite upper bounds on the cells in row i, we can treat $\mathcal{N}_i$ as an additive set by momentarily ignoring the bound imposed by N. In this case, we can cheaply identify the minimal generating set (with respect to additivity) of $\mathcal{N}_i$ and treat each generator as a separate term in the knapsack constraint. We implemented such an approach and found that its performance was similar to the method described next. However, the method below has the significant advantage of working just as well when *a priori* upper cell bounds are imposed or even when an explicit list of possible cell counts is specified beforehand.

To replace the dynamic programming approach, note that it suffices to calculate a sum of sets $\sum_i \mathcal{N}_i$ involving summands of cardinality on the order of N, but restricting the result so that we exclude elements exceeding N. Conceptually, we start with a set $\mathcal{S} = \{0\}$, then sequentially update $\mathcal{S}$ by including sums of elements of a set $\mathcal{N}_i$ with elements of $\mathcal{S}$, truncating the result to remove elements larger than N as they occur. For this purpose, we store $\mathcal{S}$ and $\mathcal{N}_i$ as indicator vectors with entries $s_k \in \{0, 1\}$ for $k = 0, 1, \ldots, N$. The convolution of these indicators is the vector whose entry at index $k = 0, 1, \ldots, 2N$ gives the multiplicity of the pairwise sums from $\mathcal{S} + \mathcal{N}_i$ that result in a value of k. In particular, the support of this convolution is the desired sum $\mathcal{S} + \mathcal{N}_i$. It is well known that we can compute the convolution efficiently as follows (Loan, 1992): pad both indicators with N additional zeros for indices $k > N$, calculate the discrete Fourier transform of the padded indicators, take the entry-wise product of the transformed vectors, and then calculate the inverse transform of that product to get the desired convolution. With fast Fourier transforms, the convolution requires only $O(N \log N)$ operations. Once the convolution has been obtained, its nonzero (rounded) entries for indices $k = 0, \ldots, N$ constitute the support of the updated $\mathcal{S}$.

Next, note that most of the work needed to calculate the summation $\sum_{i' \neq i} \mathcal{N}_{i'}$ for a given value of i is repeated for many other values of i, because only one value of i' is omitted from each summation. We exploit this redundancy by separately building the partial accumulations $\sum_{i' < i} \mathcal{N}_{i'}$ and $\sum_{i' > i} \mathcal{N}_{i'}$. Specifically, we compute and store the values of (say) the former for all i in a single forward pass and then backtrack with a simple update of a single copy of the latter. If the required storage size $(I + 1)N$ is too great to fit into computer memory then we employ the following divide-and-conquer scheme, which is similar to that presented by Wright and Smucker (2014) for their dynamic programming approach.

In this algorithm, the collection of row indices $\{1, \ldots, I\}$ is iteratively bisected to create a binary tree of nested partitions. The values of L_d and U_d give the ranges of row indices considered in each iteration of the while-loop. The subscript d denotes the depth within the tree and $\delta \in \{\pm 1\}$ is the direction of motion (deeper or shallower) as the tree is traversed. A sum $\mathcal{S}$ of sets $\mathcal{N}_i$ over some subset of rows is calculated at each stage of the bisection and a list of selected such sums $\mathcal{S}_d$ is stored for subsequent reuse. The core task of updating $\mathcal{S}$ in part (c) of step 2 is performed $O(I \log I)$ times; the contribution to $\mathcal{S}$ of part (c) in step 3 can kept separate until needed and therefore only performed I times. Using this arrangement of the work, the entire second phase of the decomposition needs $O((I \log I)(N \log N))$ operations.

Algorithm 1 Set $d := 1$, $\delta := 1$, $L_1 := 1$, $U_1 := I$, and $\mathcal{S} := \{0\}$. While $d > 0$ do the following:

1. If $L_d = U_d$, then:
   (a) Set $i := L_d$ and $\mathcal{N}_i^{\text{feas}} := \mathcal{N}_i \cap (N - \mathcal{S})$.
   (b) Set $\delta := -1$ and go to step 4.
2. If $\delta = 1$, then:
   (a) Set $L_{d+1} := \lceil (U_d + L_d)/2 \rceil$ and $U_{d+1} := U_d$.
   (b) If $d > 1$, then set $\mathcal{S}_{d-1} := \mathcal{S}$.
   (c) For $i := L_d, \ldots, L_{d+1} - 1$, set $\mathcal{S} := (\mathcal{S} + \mathcal{N}_i) \cap \{0, \ldots, N\}$.
   (d) Go to step 4.
3. If $L_d < L_{d+1}$, then:
   (a) Set $U_{d+1} := L_{d+1} - 1$ and $L_{d+1} := L_d$.
   (b) If $d > 1$, then set $\mathcal{S} := \mathcal{S}_{d-1}$. Otherwise, set $\mathcal{S} := \{0\}$.
   (c) For $i := U_{d+1} + 1, \ldots, U_d$, set $\mathcal{S} := (\mathcal{S} + \mathcal{N}_i) \cap \{0, \ldots, N\}$.
   (d) Set $\delta := 1$ and go to step 4.
4. Set $d := d + \delta$.

Here are few more comments comparing the above with the approach taken by Wright and Smucker (2014) for calculating cell bounds based on *unrounded* conditionals. Exploiting the particularly simple form of $\mathcal{N}_i$ in that setting, they presented a dynamic programming algorithm in which the update of $\mathcal{S}$ uses $O(N)$ operations instead of the $O(N \log N)$ needed by the update described here. Moreover, the first and third phases listed in Section 2.2 of the present paper can be replaced by almost trivial calculations in the unrounded case. Therefore, it generally takes at least one order of magnitude longer (sometimes much longer) to solve the cell-bounding problem for rounded conditionals than for unrounded conditionals.

We end this section with a note about an important special case, namely, that in which some row $\bar{\imath}$ has a lone nonzero entry (necessarily unity for the table of conditionals) and no *a priori* bounds are associated with that row. This situation was also singled out by Wright and Smucker (2013) 2014). For such a special row $\bar{\imath}$, the set $\mathcal{N}_{\bar{\imath}}$ consists of all integers from 1 through N. This considerably simplifies the calculation of $\mathcal{N}_i^{\text{feas}}$ for all $i \neq \bar{\imath}$, because it implies that $\sum_{i' \neq i} \mathcal{N}_{i'}$ consists of all integers from $\sum_{i' \neq i} \min \mathcal{N}_{i'}$ through $\sum_{i' \neq i} \max \mathcal{N}_{i'}$. In other words, $\mathcal{N}_i^{\text{feas}}$ is simply the restriction of $\mathcal{N}_{\bar{\imath}}$ to those elements ranging from $N - \sum_{i' \neq i} \max \mathcal{N}_{i'}$ through $N - \sum_{i' \neq i} \min \mathcal{N}_{i'}$. Consequently, we see that the convolution-based calculation of $\sum_{i' \neq i} \mathcal{N}_{i'}$ is needed only when $i = \bar{\imath}$, and that too can be avoided if there is a second such row. A further ramification is that tables with one or more such rows tend to have very wide bounds, which is good news for confidentiality but perhaps bad news for statistical utility.

3. Examples

This section presents a few examples to highlight the difference in information provided by rounded versus unrounded conditionals. Throughout the section, we only consider tables in which the rounding or perturbation is *consistent*, meaning that the decimal values of the approximate conditionals sum to unity within each row. However, consistent rounding is not required for the decomposition procedure introduced in Section 2.2.

First we consider the artificial Table 1 introduced in Section 1, for which some rounded conditionals are shown in Table 3. These correspond to the standard nearest-value rounding, which leads to the unique choice for 3-digit rounding shown in Table 3(b) and two possibilities for 2-digit rounding, as shown by the second row in Tables 3(c) and 3(d).

Table 3
Row conditionals of Table 1(c) rewritten in decimal form and rounded.

(a) $\epsilon = 0$ (repeating)		(b) $\epsilon = 0.0005$		(c) $\epsilon = 0.005$		(d) $\epsilon = 0.005$	
$0.\overline{428571}$	$0.\overline{571428}$	0.429	0.571	0.43	0.57	0.43	0.57
$0.625\overline{0}$	$0.375\overline{0}$	0.625	0.375	0.63	0.37	0.62	0.38
$0.4\overline{0}$	$0.6\overline{0}$	0.400	0.600	0.40	0.60	0.40	0.60
$0.\overline{5}$	$0.\overline{4}$	0.556	0.444	0.56	0.44	0.56	0.44

When determining possible cell counts based on the three-digit values in Table 3(b), the first phase of the method of Section 2.2 identifies each $\mathcal{N}_i$ as consisting of all multiples (no larger than $N = 48$) of the following values: 7 for the first row, 8 for the second, and 5 and 9 for the third and fourth rows, respectively. Note that these are also the corresponding denominators shown in Table 1(c). Next, the second phase restricts the possibilities to these row sums:

$$\mathcal{N}_1^{\text{feas}} = \{7, 21\}, \quad \mathcal{N}_2^{\text{feas}} = \{8\}, \quad \mathcal{N}_3^{\text{feas}} = \{10, 15\}, \quad \mathcal{N}_4^{\text{feas}} = \{9, 18\}.$$

The third phase then determines that Tables 1(a)–(b) are the only possibilities, and so the 3-digit rounded conditionals yield the same disclosure risk as the unrounded conditionals. In particular, the true counts in the second row are disclosed by this 3-digit rounding.

Now suppose the same calculations are performed under the slightly looser proximity condition having $\epsilon = 0.001$, for which there are 36 possible tables of 3-digit conditionals corresponding to Table 3(a). It turns out that all 36 lead to exactly the same results as above in each phase of the decomposition method. Consequently, a 3-digit release of these conditionals must include perturbations with $\epsilon > 0.001$ if we wish to avoid exposing the small count of 3 in the second column.

Taking a large step in that direction, we consider releasing either of the two-digit Tables 3(c) and 3(d). Applying the first phase of the solution procedure to these yields

$$\begin{aligned}
\mathcal{N}_1 &= \{\mathbf{7}, \mathbf{14}, 16, 19, \mathbf{21}, 23, 25, 26, \mathbf{28}, 30, 32, 33, \mathbf{35}, 37, 38, \\
&\quad 39, 40, 41, \mathbf{42}, 44, 45, 46, 47, 48\}, \\
\mathcal{N}_2 &= \{\mathbf{8}, \mathbf{16}, \mathbf{24}, 27, 29, \mathbf{32}, 35, 36, 37, \mathbf{40}, 43, 44, 45, 47, \mathbf{48}\} \\
&\quad \cup \begin{cases} \{11, 19, 22, 25, 30, 33, 38, 41, 46\}, & \text{Table 3(c)}, \\ \{13, 18, 21, 26, 31, 34, 39, 42\}, & \text{Table 3(d)}, \end{cases} \\
\mathcal{N}_3 &= \{\mathbf{5}, \mathbf{10}, \mathbf{15}, \mathbf{20}, 22, 23, \mathbf{25}, 27, 28, \mathbf{30}, 32, 33, \mathbf{35}, 37, 38, \\
&\quad \mathbf{40}, 41, 42, 43, 44, \mathbf{45}, 46, 47, 48\}, \\
\mathcal{N}_4 &= \{\mathbf{9}, 16, \mathbf{18}, 20, 23, 25, \mathbf{27}, 29, 30, 32, 34, \mathbf{36}, 37, 38, 39, \\
&\quad 40, 41, 43, 44, \mathbf{45}, 46, 47, 48\},
\end{aligned}$$

where the numbers in bold type indicate the values obtained previously in the 3-digit case (to facilitate a comparison). The second phase of the decomposition restricts these to

$$\mathcal{N}_1^{\text{feas}} = \{\mathbf{7}, 14, 16, 19, \mathbf{21}, 26\} \cup \begin{cases} \{23\}, & \text{Table 3(c)}, \\ \emptyset, & \text{Table 3(d)}, \end{cases}$$

$$\mathcal{N}_2^{\text{feas}} = \{\mathbf{8}, 16, 27\} \cup \begin{cases} \{11, 22\}, & \text{Table 3(c)}, \\ \{13\}, & \text{Table 3(d)}, \end{cases}$$

$$\mathcal{N}_3^{\text{feas}} = \{5, \mathbf{10}, \mathbf{15}\},$$

Table 4
Possible cell counts based on rounded conditionals in Tables 3(c) and 3(d).

$\{\mathbf{3}, 6, 7, 8, \mathbf{9}, 11\} \cup \begin{cases} \{10\}, & \text{Table 3(c)} \\ \emptyset, & \text{Table 3(d)} \end{cases}$	$\{\mathbf{4}, 8, 9, 11, \mathbf{12}, 15\} \cup \begin{cases} \{13\}, & \text{Table 3(c)} \\ \emptyset, & \text{Table 3(d)} \end{cases}$
$\{\mathbf{5}, 10, 17\} \cup \begin{cases} \{7, 14\}, & \text{Table 3(c)} \\ \{8, 11\}, & \text{Table 3(d)} \end{cases}$	$\{\mathbf{3}, 6, 10\} \cup \begin{cases} \{4, 8\}, & \text{Table 3(c)} \\ \{5, 7\}, & \text{Table 3(d)} \end{cases}$
$\{2, \mathbf{4}, \mathbf{6}\}$	$\{3, \mathbf{6}, \mathbf{9}\}$
$\{\mathbf{5}, 9, \mathbf{10}, 11, 13\} \cup \begin{cases} \{14\}, & \text{Table 3(c)} \\ \emptyset, & \text{Table 3(d)} \end{cases}$	$\{\mathbf{4}, 7, \mathbf{8}, 9, 10\} \cup \begin{cases} \{11\}, & \text{Table 3(c)} \\ \emptyset, & \text{Table 3(d)} \end{cases}$

Table 5
4×4 contingency table with $N = 135$.

15	1	3	1	20
20	10	10	15	55
3	10	10	2	25
12	14	7	2	35

Table 6
Unrounded row conditionals for Table 5.

$3/4 = 0.75\overline{0}$	$1/20 = 0.05\overline{0}$	$3/20 = 0.15\overline{0}$	$1/20 = 0.05\overline{0}$
$4/11 = 0.\overline{36}$	$2/11 = 0.\overline{18}$	$2/11 = 0.\overline{18}$	$3/11 = 0.\overline{27}$
$3/25 = 0.12\overline{0}$	$2/5 = 0.4\overline{0}$	$2/5 = 0.4\overline{0}$	$2/25 = 0.04\overline{0}$
$12/35 = 0.3\overline{428571}$	$2/5 = 0.4\overline{0}$	$1/5 = 0.2\overline{0}$	$2/35 = 0.0\overline{571428}$

Table 7
Rounded conditionals for Table 5.

(a) $\epsilon = 0.001$				(b) $\epsilon = 0.01$			
0.750	0.050	0.150	0.050	0.75	0.05	0.15	0.05
0.364	0.182	0.181	0.273	0.37	0.18	0.18	0.27
0.120	0.400	0.400	0.040	0.12	0.40	0.40	0.08
0.343	0.400	0.200	0.057	0.34	0.40	0.20	0.06

Table 8
Cell bounds based on Table 7(b).

[15,63]	[1,5]	[3,13]	[1,5]	[20,84]
[4,28]	[2,14]	[2,14]	[3,21]	[11,75]
[3,11]	[10,36]	[10,36]	[2,8]	[25,89]
[5,27]	[6,32]	[3,16]	[1,5]	[15,79]

$$\mathcal{N}_4^{\text{feas}} = \{\mathbf{9}, 16, \mathbf{18}, 20, 23\} \cup \begin{cases} \{25\}, & \text{Table 3(c)}, \\ \emptyset, & \text{Table 3(d)}. \end{cases}$$

The corresponding lists of numerators obtained in the third phase are shown in Table 4. The results of all three phases are unchanged if we process Table 3(c and d) using $\epsilon = 0.01$ instead of 0.005.

The next example illustrates an even wider gap between 2-digit and 3-digit roundings, along with a few other noteworthy possibilities. Table 5, taken from Report on statistical disclosure limitation methodology (2005), was shown by Slavković (2004) to be the unique 4×4 contingency table of sample size $N = 135$ having the unrounded row conditionals given in Table 6. In other words, all cell counts are revealed by these unrounded conditionals and the many small values might pose a serious disclosure risk. Slavković and Lee (2010) used a particular one-digit rounding of Table 6 for the purpose of creating and enumerating synthetic tables of counts preserving those rounded values (as if they were the true conditionals). Their findings suggested no compromise in data utility for the use of standard statistical hypothesis tests of independence. Here we focus on how disclosure is affected by the rounding precision.

Two sets of rounded conditionals are shown in Table 7(a and b), where all entries were calculated by first using nearest-value rounding (to 3 and 2 digits, respectively) and then adjusting one entry in the second row so that the row sum is unity. From the 3-digit rounding in Table 7(a), the decomposition completely recovers the true cell counts. On the other hand, the two-digit rounding in Table 7(b) leads to the cell bounds shown in Table 8. Many of these cell bounds are quite wide. In the six cells where the true counts are 3 or less, all integers in the bounding intervals are possible. The remaining cells can each take at least 12 distinct values, and the upper left cell can take 31 values.

Notice that each row-sum upper bound in Table 8 is less than the corresponding sum of the cell upper bounds in that row, an impossibility for bounds obtained from unrounded conditionals. This suggests a lack of monotonicity in how the numerators within a row are matched to the feasible denominators in $\mathcal{N}_i^{\text{feas}}$. In fact, it is also

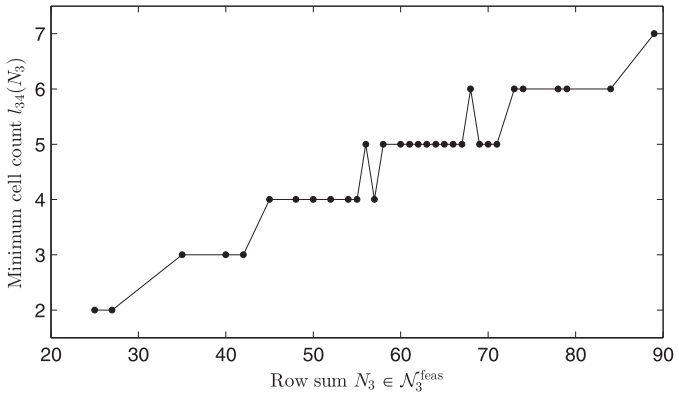

Fig. 1. Tightest lower bound on row 3, column 4 from Table 7(b) as a function of the allowable sums for row 3. This illustrates the lack of monotonicity of this function.

possible that the tightest upper (or lower) bound for an individual cell does not occur at the upper (or lower, respectively) bound for the row sum. Although this cell-wise difficulty does not affect the bounds in the present example, the underlying cell-wise non-monotonicity of numerator to denominator does occur. Fig. 1 shows how the tightest lower bound on row 3, column 4 changes with respect to the row sums in $\mathcal{N}_3^{\text{feas}}$. This is why the second phase of the decomposition procedure must return the specific allowable row sums rather than merely bounds on those sums.

4. Computational tests

In this section, we present results of some computational tests showing the running time of the proposed decomposition. Some of the tests used real-world contingency tables from the literature. In general, the original table of counts in each case is a k-way table for which each dimension denotes a discrete random variable, usually taking a small number of values. In creating a summary table of conditional probabilities, some of the random variables are designated as *predictor* variables and some as *response* variables; the former then correspond to the rows of the summary table and the latter to its columns. Aggregation is performed over any random variables omitted from those two designations. This always yields a two-way contingency table, from which the original table can be recovered directly when there is no aggregation over variables. It is also possible to aggregate over the levels taken by the discrete variables. Wright and Smucker (2014) provide a detailed example of such table reshaping and aggregation in the context of unrounded conditionals.

We constructed such two-way tables and their corresponding row conditionals from five data sets as follows.

- A 3-way table (Haberman, 1978) with a $3^2 \times 3$ summary and $N = 1055$, using dimension 3 as the response (columns) and the other two dimensions as predictors (rows).
- A 4-way table (Koch, Amara, Atkinson, & Stanish, 1983) with a $2^3 \times 3$ summary and $N = 193$, using dimension 4 as the response and the other three dimensions as predictors.
- A 6-way table (Edwards & Havranek, 1985) with a $2^5 \times 2$ summary and $N = 1841$, using dimension 6 as the response and the other five dimensions as predictors.
- An 8-way table from the U.S. Census Bureau's 1993 Current Population Survey (see, for instance, Gomatam, Karr, & Sanil, 2003; Sanil, Gomatam, Karr, & Liu, 2003) with a $(2^2 \cdot 3) \times 2$ summary and $N = 48842$. Dimension 8 was used as the response, dimensions 4, 6 and 7 as predictors, and aggregation was performed over dimensions 1, 2, 3 and 5.
- A 16-way table from the National Long Term Care Study (see, for instance, Erosheva, Fienberg, & Joutard, 2007) with a $2^3 \times 2$ summary and $N = 21574$. Dimension 4 was used as the response, dimensions 1, 2, and 3 as predictors, and aggregation was performed over the other twelve dimensions.

Table 9
Computational results for two-way contingency tables using real data.

Original table	Rounding digits	N	I	J	Running times (s)		
					Decomp.	CPLEX	
						All IPs	Asymm.
3-way	2	1055	9	3	0.0172	2.4892	0.2036
3-way	3	1055	9	3	0.0152	9.0636	1.1728
3-way	4	1055	9	3	0.0146	7.7910	0.1418
4-way	2	193	8	3	0.0032	0.8832	0.1670
4-way	3	193	8	3	0.0030	2.2901	0.4239
6-way	2	1841	32	2	0.0993	7.2090	0.3500
6-way	3	1841	32	2	0.0991	5.5241	0.4926
6-way	4	1841	32	2	0.0962	16.1957	0.7247
8-way	3	48842	12	2	0.4126	3.7203	-
8-way	4	48842	12	2	0.4097	5.0515	-
8-way	5	48842	12	2	0.4208	13.3775	-
16-way	3	21574	8	2	0.1762	5.1954	-
16-way	4	21574	8	2	0.1644	8.6395	-
16-way	5	21574	8	2	0.1677	7.8118	-

Although the six-way table includes a row with a lone nonzero entry (as discussed at the end of Section 2.2), we did not exploit that opportunity because the computational tests focused on how decomposition time varies with problem dimension and rounding precision. Likewise, larger and less-aggregated rearrangements of the 8- and 16-way tables tend to have many of those special rows and were therefore omitted here because they are handled much more efficiently by the lone-nonzero procedure.

In the formulations considered here, we used the trivial *a priori* bounds on all cell and row counts: $l_{ij} = L_i = 1$ and $u_{ij} = U_i = N$. Consistent roundings were employed with $\epsilon = 10^{-d}$ in (5) for d-digit roundings; note that standard roundings with $\epsilon = 0.5 \times 10^{-d}$ need not give values summing to unity within rows. Table rearrangement, problem formulation, and decomposition were all performed in the MATLAB 7.14 environment. Details about the test instances are given in Table 9. The running times reported in Table 9 were obtained on a 2 × quad-core Intel 64-bit (3.40 GHz, 8GB RAM) platform with the Windows 7 Enterprise operating system. For a given table of conditionals, the decomposition requires essentially the same amount of effort regardless of the rounding precision. Recall from Section 2.2 that we expect the operation count to be dominated by $O((I \ln I)(N \ln N))$. Fig. 2 shows a scatterplot of the running times for each of the contingency tables along with the curve $t = [(I \ln I)(N \ln N)]^{0.6} \cdot 5.25 \times 10^{-5}$. This suggests that the bound in Section 2.2 is perhaps overly pessimistic, at least for tables with just a few dozen rows.

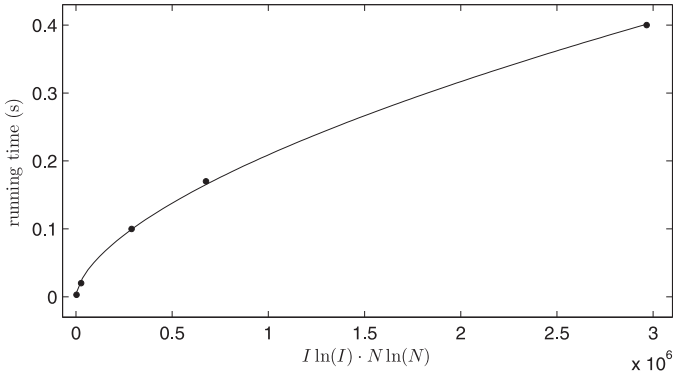

Fig. 2. Actual runtime versus the predicted bound on runtime from Section 2.2, shown for the problem instances of Table 9.

Table 10
Computational results for randomly generated two-way contingency tables.

I	J	Symm.-breaking constraints				N		Running times (s)	
		Cell		Row					
		Min	Max	Min	Max	Min	Max	Decomp.	CPLEX
32	2	0	2	8	10	753	901	0.0587	0.3820
64	2	1	4	28	34	1555	1786	0.1567	0.6582
128	2	4	9	86	88	3164	3464	0.3664	1.2401
256	2	6	15	209	211	6432	6808	1.1979	2.2869
512	2	10	26	465	465	13019	13694	4.5498	9.4638
1024	2	40	42	977	977	26129	27235	25.6178	29.2171
32	4	5	7	0	1	1555	1786	0.0959	1.4165
64	4	6	17	0	2	3164	3464	0.4392	8.6405
128	4	17	30	1	5	6432	6808	1.8889	16.0226
256	4	37	47	6	16	13019	13694	9.8843	80.9988
512	4	78	97	31	44	26129	27235	48.7597	318.5286
32	8	29	38	0	0	3164	3464	0.2275	49.7171
64	8	63	75	0	0	6432	6808	1.0462	148.7338
128	8	121	158	0	0	13019	13694	5.2797	725.1805
256	8	270	292	0	2	26129	27235	22.8295	> 1000
2	32	48	50	0	0	753	901	0.0198	0.8903
2	64	119	120	0	0	1555	1786	0.0389	1.0454
2	128	250	250	0	0	3164	3464	0.1031	1.6506
2	256	508	508	1	1	6432	6808	0.1276	0.9752
2	512	1020	1020	1	1	13019	13694	0.3974	2.8281
2	1024	2044	2044	1	1	26129	27235	1.3416	4.6180
4	32	98	103	0	0	1555	1786	0.0455	5.9267
4	64	239	240	0	1	3164	3464	0.1243	10.9305
4	128	500	501	0	1	6432	6808	0.2697	8.0010
4	256	1016	1016	3	3	13019	13694	0.3123	3.4285
4	512	2040	2040	3	3	26129	27235	1.0465	7.4883
8	32	198	202	0	0	3164	3464	0.1560	81.0652[a]
8	64	476	479	0	2	6432	6808	0.3451	24.0977[a]
8	128	1000	1000	1	3	13019	13694	0.9155	88.6950
8	256	2032	2032	7	7	26129	27235	0.9197	12.7754

[a] Mean excludes two instances requiring over 1000s.

For the purpose of a very simplistic comparison, each cell-bounding problem (1)–(6) was also solved by a separate call to CPLEX 12.5 (with default settings), a popular commercial software package for integer and linear optimization. As shown in the columns labeled "decomp." and "all IPs" of Table 9, the decomposition outperforms this direct application of CPLEX by at least an order of magnitude in almost every case. The CPLEX times also show relatively little dependence on sample size. The number of branch-and-bound nodes per cell optimization ranged from 0 (i.e., root node only) to 10413. For some cells, even the optimizations involving only the root node occasionally made heavy use of the default CPLEX preprocessing and cut-generation to obtain an integer solution. Careful tuning of CPLEX parameters and workflow might well overcome these difficulties; see Section 5 for some possibilities. Nevertheless, calling a general-purpose solver has drawbacks such as the difficulty of obtaining all possible cell counts (in addition to the extremes) and a potential lack of portability. On the other hand, the proposed decomposition approach consists of fewer than 150 lines of MATLAB code and can therefore be ported to a variety of programming environments and easily modified to address related questions.

Contingency tables lacking *a priori* bounds on individual cell or row counts sometimes exhibit symmetry insofar as multiple cells within a row have the same count or multiple rows have the same (sorted) list of cell counts. In such cases, we need only solve a subset of the full collection of IJ integer programming problems. Moreover, we can improve the performance of branch-and-bound in CPLEX by introducing symmetry-breaking constraints, such as forcing $n_{ij} \leq n_{ik}$ if cells j and k (for $j < k$) within row i are symmetrically equivalent. The final column in Table 9, labeled "asymm.", gives the running times when such considerations are taken into account. This shows that CPLEX can perform much better when the symmetry aspects are removed, but it is still an order of magnitude slower than the proposed decomposition on these small tables. The tables derived from the 3-way and 4-way examples have several repeated cell counts whereas those based on the 6-way example also have some repeated rows. For these instances, many more of the cell optimizations involved only the root node (alongside cuts and preprocessing) and the largest node-count was reduced to 467. On the other hand, the given arrangements of the 8-way and 16-way tables exhibit no such symmetry and therefore no improvements were possible in those cases.

The contingency tables considered so far are relatively small examples, so it is worth investigating algorithm performance on a somewhat larger scale. Although the 8-way and 16-way tables above can be rearranged to create tables with hundreds or thousands of rows and much symmetry, such rearrangements generally have several rows with lone nonzero entries and can be handled trivially (as discussed at the end of Section 2.2). We therefore consider randomly generated tables instead. For each of several 2-way table dimensions I and J, we generated five instances with integer cell counts randomly selected from 1 through 25 and built tables of conditionals with 2-digit rounding. All of these instances exhibited quite a lot of symmetry, so the improvements discussed in the previous paragraph were again employed. The results are shown in Table 10, where we also indicate the number of symmetry-breaking constraints available and employed for the various instances. The gap between the proposed decomposition and CPLEX is narrow for contingency tables with two columns and many repeated rows, but it becomes substantially wider as the number of columns is increased and the symmetry moves from being mainly within rows rather than among rows. At the other extreme, in which the number of rows is far smaller than the number of columns, the gap is again less pronounced. In those instances, most of the computing time for the proposed method is spent on the closed-form column-wise calculations (7)–(10) of phases 1 and 3. Note that

Table 11
Computational results for larger randomly generated contingency tables.

I	J	Symm.-breaking Constraints		N	Running times (s)	
		Cell	Row		Decomp.	CPLEX
16	8	14	0	31727	0.6398	23.3657
32	8	28	0	59438	2.4658	70.9055
64	8	42	0	119546	13.0437	230.1883
128	8	124	0	238027	59.4213	612.7021
256	8	228	0	489350	309.7784	> 10000
8	16	50	0	31727	0.4315	26.2959
8	32	192	0	59438	1.1165	> 1000
8	64	471	0	119546	3.7923	139.1816
8	128	996	1	238027	14.6367	79.2860
8	256	2026	4	489350	41.1244	97.2863
8	8	6	0	16701	0.4796	11.6754
16	16	101	0	59438	1.9538	70.8363
32	32	776	0	238027	28.1429	> 1000
64	64	3786	5	993210	912.1266	> 10000
128	128	15931	85	3923348	3388.4126	> 36000
256	256	64839	249	15652394	18557.4222	> 80000

such instances have much greater within-row symmetry because the cell counts are capped at 25.

This results in Table 10 raise further questions about how the proposed method performs when many cell counts are not small and the sample size is very large, or when the contingency table is square in shape. To test these, we generated larger instances in which only 5% of cells are given counts in [0, 4] and the remaining 95% have counts uniformly distributed in [5, 499], again with conditionals given by two-digit rounding. Sample sizes for such tables can be in the hundreds of thousands or millions. Because such instances are large and time-consuming to solve by either method, only one instance was generated for each of the configuration sizes considered. The results are shown in Table 11. As with the medium-sized tables, the gap between the proposed method and the naive use of CPLEX is often an order of magnitude or more. Again, this suggests that the fruitful application of a commercial solver might require parameter tuning.

We end this section by reminding the reader that none of the computational tests reported here involve *a priori* bounds on cell counts or row sums. When such bounds are imposed by knowledge of another table of different shape and arrangement based on the same underlying data, the symmetry can disappear entirely. In that case, the proposed method requires (by its nature) even less time to solve the cell-bounding problems. This matter is a topic for future research, given that simultaneous consideration of both tables together is of greater interest than the effect that bounds from two tables have upon each other separately.

5. Discussion

We have proposed an optimization model and solution framework for determining the tightest bounds (indeed, all possible values) for each cell of a two-way contingency table, as implied by a corresponding table of rounded conditional frequencies. The method is fast enough on tables of moderate size, even with fairly large sample sizes, to allow real-time cell-bounding during the redesign and/or aggregation of a multi-way table as a two-way table of conditionals.

To include this technique in the larger toolbox of statistical disclosure limitation will require finding ways to combine the information in a table of conditionals with that from other releases of the same underlying table. The bounds and possible counts found by our methods might be used as the starting point for procedures that obtain cell bounds or cell counts on the basis of information about marginal counts or tables with adjusted cell counts (see, for example, Buzzigoli & Gusti, 1998; Cox, 2002; Dobra & Fienberg, 2010, Slavković, Zhu, & Petrović, Dobra, 2012).

There is also potential scope for generalizing the approach presented here to incorporate additional knowledge about the underlying contingency table. In the setting of unrounded conditionals Wright and Smucker (2013) note that, by means of simple and inexpensive preprocessing, their formulation can easily address bounds on partial sums of the cell counts that lie within a row (but not on sums of cells that lie in different rows). This is an effective way to use information released in the form of one or more contingency tables that aggregate counts from the underlying table. Adding the same functionality to the method of Section 2.2 faces two main obstacles. One is the potential lack of easily checked conditions like (8)–(10), which may need to be replaced by $O(IJN)$ integer optimizations to address more general constraints on sums of cells within rows. The second difficulty is that we might lose the guarantee (as in the final sentence of Theorem 2.1) that *all* integer values of n_{ij} between the extreme values are attainable. For these reasons, determining all possible cell counts could require far more computational effort than the procedure given herein. However, it may be possible to handle specially structured collections of constraints by directly adapting the proof of Theorem 2.1. This is an area of current investigation.

As with any new integer programming problem, there are potential opportunities for exploring the application of more traditional operations research techniques. For example, it is likely possible to tune the use of a general-purpose solver for much greater efficiency. As noted by an anonymous referee, standard solver-specific options that might prove advantageous include local branching heuristics, relaxing the MIP optimality gap, forcing use of the feasibility pump (CPLEX automatically decides on this, by default), and better sharing of MIP information among cell problems. In this way, a general-purpose solver might even be able to solve much larger problem instances. However, improving the efficiency of a general-purpose solver does not address its inability to identify actual possible cell counts instead of merely producing the extreme counts.

Acknowledgments

The authors thank Professor Byran J. Smucker for introducing us to this problem and for helpful comments during the preparation of the paper. We are also grateful to three anonymous referees who provided many suggestions for improving the paper. Miami University's "RedHawk" computing cluster was used for preliminary investigations. This research was partially supported by an Assigned Research Appointment for the second author at Miami University during the spring of 2013. The first author's contributions to this work are drawn from his Master degree project in the Department of Mathematics at Miami University.

References

Almeida, M. T., & Carvalho, F. D. (2005). Exact disclosure prevention in two-dimensional statistical tables. *Computers and Operations Research, 32*, 2919–2936.
Buzzigoli, L., & Gusti, A. (1998). An algorithm to calculate the upper and lower bounds of the elements of an array given its marginals. In *Statistical Data Protection (SDP 1998) Proceedings* (pp. 131–147). Luxembourg: Eurostat.
Castro, J. (2006). Minimum-distance controlled perturbation methods for large-scale tabular data protection. *European Journal of Operational Research, 171*, 39–52.
Castro, J. (2011). Extending controlled tabular adjustment for non-additive tabular data with negative protection levels. *SORT, 35*, 3–19.
Castro, J. (2012). Recent advances in optimization techniques for statistical tabular data protection. *European Journal of Operational Research, 216*(2), 257–269.
Cox, L. (1995). Network models for complementary cell suppression. *Journal of the American Statistical Association, 90*(432), 1453–1462.
Cox, L. (2002). Bounds on entries in 3-dimensional contingency tables. In J. Domingo-Ferrer (Ed.), *Inference Control in Statistical Databases – From Theory to Practice. LNCS: 2316* (pp. 21–33). Springer-Verlag.
Cox, L., & Ernst, L. (1982). Controlled rounding. *INFOR, 20*, 423–432.
Dobra, A. (2012). Dynamic markov bases. *Journal of Computational and Graphical Statistics, 21*, 496–517.

Dobra, A., & Fienberg, S. E. (2010). The generalized shuttle algorithm. In M. P. Rogantin, & H. P. Wynn (Eds.), *Algebraic and Geometric Methods in Statistics* (pp. 135–156). Cambridge University Press.

Edwards, D., & Havranek, T. (1985). A fast procedure for model search in multidimensional contingency tables. *Biometrika, 72*, 339–351.

Erosheva, E., Fienberg, S., & Joutard, C. (2007). Describing disability through individual-level mixture models for multivariate binary data. *Annals of Applied Statistics, 1*, 346–384.

Fienberg, S. E., & Slavković, A. B. (2005). Preserving the confidentiality of categorical statistical data bases when releasing information for association rules. *Data Mining and Knowledge Discovery, 11*, 155–180.

Fischetti, M., & Salazar-González, J. J. (1999). Models and algorithms for the 2-dimensional cell suppression problem in statistical disclosure control. *Mathematical Programming, 84*, 283–312.

Gomatam, S., Karr, A. F., & Sanil, A. (2003). A risk-utility framework for categorical data swapping. *Technical report 132*. National Institute of Statistical Sciences.

Haberman, S. J. (1978). *The Analysis of Qualitative Data, volume 1, 2*. Orlando: Academic Press.

Hernández-García, M. S., & Salazar-González, J. J. (2014). Enhanced controlled tabular adjustment. *Computers and Operations Research, 43*, 61–67.

Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E. S., Spicer, K., & de Wolf, P.-P. (2012). *Statistical Disclosure Control*. Chichester: John Wiley & Sons.

Kelly, J. P., Golden, B. L., & Assad, A. A. (1990). Using simulated annealing to solve controlled rounding problems. *ORSA Journal on Computing, 2*, 174–185.

Kelly, J. P., Golden, B. L., Assad, A. A., & Baker, E. K. (1990). Controlled rounding of tabular data. *Operations Research, 38*, 760–772.

Koch, G., Amara, J., Atkinson, S., & Stanish, W. (1983). Overview of categorical analysis methods. *SAS-SUGI, 8*, 785–795.

Loan, C. Van (1992). *Computational Frameworks for the Fast Fourier Transform*. Philadelphia, PA: SIAM.

Muralidhar, K., & Sarathy, R. (2006). Data shuffling: a new masking approach for numerical data. *Management Science, 52*, 658–670.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley-Interscience.

Report on statistical disclosure limitation methodology. 2005. Federal Committee on Statistical Methodology, Statistical Policy Working Paper 22 (Version Two).

Salazar-González, J.-J. (2004). Mathematical models for applying cell suppression methodology in statistical data protection. *European Journal of Operational Research, 154*, 740–754.

Salazar-González, J.-J. (2005). Framework for different statistical disclosure limitation methods. *Operations Research, 53*, 819–829.

Salazar-González, J.-J. (2006). Controlled rounding and cell perturbation: Statistical disclosure limitation methods for tabular data. *Mathematical Programming, 105*(2-3), 583–603.

Salazar-González, J.-J. (2008). Statistical confidentiality: Optimization techniques to protect tables. *Computers and Operations Research, 35*, 1638–1651.

Sanil, A., Gomatam, S., Karr, A. F., & Liu, C. (2003). Niss webswap: A web service for data swapping. *Journal of Statistical Software, 8*.

Slavković, A. B. (2004). *Statistical disclosure limitation beyond the margins: Characterization of joint distributions for contingency tables*. Carnegie Mellon University Phd thesis.

Slavković, A. B. (2010). Partial information releases for confidential contingency table entries: Present and future research efforts. *Journal of Privacy and Confidentiality, 1*(2), 253–264.

Slavković, A. B., & Lee, J. (2010). Synthetic two-way contingency tables that preserve conditionals frequencies. *Statistical Methodology, 7*, 225–239.

Slavković, A. B., Zhu, X., & Petrović, S. (2015). Fibers of multi-way contingency tables given conditionals: relation to marginals, cell bounds and markov bases. *Annals of the Institute of Mathematical Statistics, 62*(4), 621–648.

Smucker, B., & Slavković, A. B. (2008). Cell bounds in two-way contingency tables based on conditional frequencies. In J. Domingo-Ferrer, & Y. Saygin (Eds.), *PSD 2008. LNCS: 5262* (pp. 64–76). Berlin Heidelberg: Springer-Verlag.

Smucker, B. J., Slavković, A. B., & Zhu, X. (2012). Cell bounds in k-way tables given conditional frequencies. *Journal of Official Statistics, 28*(1), 121–140.

Wright, S. E., & Smucker, B. J. (2013). An intuitive formulation and solution of the exact cell-bounding problem for contingency tables of conditional frequencies. *Journal of Privacy and Confidentiality, 5*, 133–156.

Wright, S. E., & Smucker, B. J. (2014). Rapid calculation of exact cell bounds for contingency tables from conditional frequencies. *Computers and Operations Research, 52*, 113–122.