Efficient Detection of Leaked Information in Cross Tabulated Tables: Linear Invariant Test

By

Ming-Yang Kao
Department of Computer Science
Indiana University
Bloomington, IN 47405

and

Dan Gusfield
Department of Electrical and Computer Engineering
University of California
Davis, CA 95616

TECHNICAL REPORT NO. 255

Efficient Detection of Leaked Information
in Cross Tabulated Tables: Linear Invariant Test

by

Ming-Yang Kao and Dan Gusfield

July, 1988

Most of this work was done while the authors were at the Department of Computer Science, Yale University. The first author was supported in part by grants from NSF and ONR (grant numbers to be filled). The second author was supported in part by grants from NSF (MCS-81-05894) and the U.S. Census Bureau (JSA 86-9).

Abstract

Cross tabulated tables are often published with some cells suppressed to protect sensitive information. A suppressed cell is unprotected if it has a unique feasible value; similarly, a linear combination of suppressed cells is unprotected if it has a unique feasible value. An unprotected linear combination is called a *linear invariant.* This paper presents a simple linear-time sequential algorithm to test whether any given linear combination is an invariant; this algorithm can be implemented in polylog parallel time using a polynomial number of processors on a PRAM. The results in this paper will also be used in subsequent papers for further study on other aspects of detecting and protecting information in cross tabulated tables.

1 Introduction

Cross tabulated tables are useful tools to organize and display information; in fact, they are routinely used to report statistical data. For protecting sensitive information in a table, it is a usual practice to suppress the precise values of certain sensitive cells. There are two fundamental issues concerning the effectiveness of this practice. The *detection* issue is to decide whether adversaries can deduce any significant information about the suppressed cells from the published data of the table. The *protection* issue is to study how table makers can suppress a small number of cells in addition to the sensitive ones such that the resulting table does not leak any significant information. A number of statisticians began research into these two issues by applying linear programming techniques to various problems [San84] [BCS83] [Cox80] [CS79] [Cox78] [San78] [Sante] [Cox77] [San77] [Cox75]; some of their results are briefly discussed by Denning [Den82]. Gusfield used a graph theoretic approach to detect all unprotected suppressed cells in linear time for any tables [Gus88], and to protect these cells in linear time for a large class of tables [Gus87]. Kao [Kao86] conducted the first systematic study of the area by introducing a linear algebraic approach to complement the graph theoretic one. The latter two approaches are discussed in detail in this paper; the linear programming approach is briefly compared with the latter two in the following discussion.

This paper studies two dimensional tables which publish the following three types of data: (1) the precise values of all cells except a set of sensitive ones, (2) an upper bound and a lower bound for each suppressed cell, and (3) all row sums and column sums of the complete set of cells. The suppressed cells may have real values as opposed to integer values. The bounds of the suppressed cells may be reals or infinities; the lower bound of any suppressed cell is presumably less than its upper bound; the suppressed cells may have independent bounds as opposed to uniform bounds. A *bounded feasible assignment* to a table is an assignment of values to the suppressed cells such that the bounds of the suppressed cells are satisfied, and each row or column adds up to its published sum. A suppressed cell is *unprotected* if it has a unique value at all bounded feasible assignments; similarly, a linear combination of suppressed cells with real coefficients is unprotected if it has a unique value at all bounded feasible assignments. An unprotected cell is called an *invariant cell*, and an unprotected linear combination is called a *linear invariant*. Figure 1 provides examples of tables and invariants.

This paper concentrates on the detection problem of deciding whether any linear combination of suppressed cells is a linear invariant of any given table. The problem is called the *Linear Invariant Test Problem*, and it has a straightforward linear programming solution as follows. The given table is translated into a set of linear constraints such that (1) each suppressed cell is a variable, (2) each row or column sum induces an equation, and (3) the upper and lower bounds of each suppressed cell yield a pair of inequalities. To decide whether the given linear combination is a linear invariant,

row column index	a	b	c	d	e	f	g	h	i	row sum
1	9	5	1	7	1	1	5	2	3	34
2	5	9	9	5	5	9	9	9	5	65
3	6	1	9	0	9	6	5	2	5	43
4	2	1	4	7	1	5	9	5	2	36
5	1	5	4	6	5	0	0	5	9	35
6	2	3	3	4	6	5	2	2	9	36
column sum	25	24	30	29	27	26	30	25	33	

row column index	a	b	c	d	e	f	g	h	i	row sum
1			1	7	1	1	5	2	3	34
2										65
3	6	1				6	5	2	5	43
4	2	1	4	7	1			5	2	36
5	1	5	4	6	5					35
6	2	3	3	4	6	5	2	2		36
column sum	25	24	30	29	27	26	30	25	33	

The first table is the complete table; the second table is a published version of the first table with several cells suppressed. The published lower and upper bounds for the suppressed cells are 0 and 9. The cells $C_{2,c}$, $C_{3,c}$ and $C_{6,i}$ are the only invariant cells. The following linear combination is a linear invariant: $2.5C_{1,a}+1.5C_{1,b}+3.5C_{2,a}+2.5C_{2,b}+C_{2,d}+1.5C_{2,e}+3C_{2,f}+3C_{2,g}+4C_{2,h}+2C_{2,i}+2C_{3,d}+2.5C_{3,e}+2.5C_{4,f}+2.5C_{4,g}+2.5C_{5,f}+2.5C_{5,g}+3.5C_{5,h}+1.5C_{5,i}$.

Figure 1: Examples of Tables and Invariants

it suffices to treat the linear combination as an objective function and compute its maximum and minimum subject to the above constraints. The linear combination is an invariant if and only if its maximum and minimum are equal. In sequential computation, linear programming takes more than linear time [Vai87] [PS82]; in parallel computation, linear programming is log-space complete for P [GSS82] [DLR79], so it is unlikely to be parallelizable into NC algorithms [Coo85].

In contrast to the linear programming approach, a main result of this paper shows that the Linear Invariant Test Problem can be solved in *linear* sequential time and in *polylog* parallel time using a polynomial number of processors on a PRAM. This main result is a simple corollary of a theorem that will be proved later, namely, the *Strongly Connected Basis Theorem.* This theorem is a fundamental result of an approach that combines graph theory and linear algebra; it allows a basis of a certain vector space of strongly connected graphs to be found very efficiently. This theorem will also be of great importance in future papers on characterizing the structure of leaked data and on protecting against leaked data rather than just detecting leaks.

One may consider the notion that a linear combination of suppressed cells is a linear invariant if and only if it has a unique value at all bounded feasible assignments that have *integer* values. However, this integrality constraint does not change the nature of the Linear Invariant Test Problem because the set of linear constraints induced by any table is *totally unimodular* [Mur76]. In other words, for any table and any linear combination of suppressed cells, the maximum and the minimum of the combination can be achieved at bounded feasible assignments that have integer values if the original values of the suppressed cells are integers.

The paper is organized as follows. Section 2 describes the linear algebraic and graph theoretic approach, states the Strongly Connected Basis Theorem, and applies the theorem to solve the Linear Invariant Test Problem. Section 3 proves the Strongly Connected Basis Theorem. Section

4 concludes the paper with a summary.

2 The Approach and the Main Results

2.1 The Linear Algebraic and Graph Theoretic Approach

The Linear Invariant Test Problem can be recast into a linear algebraic problem as follows. Let T be any table. The *bounded kernel of* T, denoted by $BK(T)$, is the real vector space consisting of all linear combinations of any $\alpha - \beta$, where α and β are any bounded feasible assignments of T.

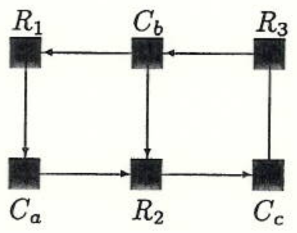
Lemma 1 *For any table T and any basis B of $BK(T)$, a linear combination f of suppressed cells in T is a linear invariant if and only if the value of f is* 0 *at all elements of B.*

Proof. Straightforward. ∎

One can use the lemma to test whether f is a linear invariant by computing the values of f merely at an arbitrary vector basis of $BK(T)$; unfortunately, it seems computationally difficult to find a basis of $BK(T)$. Finding a basis would be less difficult if there were no bounds on cell values; for this reason, two new concepts are in order. An *unbounded* feasible assignment of a table is an assignment to the suppressed cells such that the row and column sums equal their corresponding published values but the bounds of the suppressed cells may be violated. The *unbounded* kernel of T, denoted by $UK(T)$, is the real vector space consisting of all linear combinations of any $\alpha - \beta$, where α and β are any *unbounded* feasible assignments of T. The basis finding problem is closely related to that of deciding whether $BK(T) = UK(T)$, which in turn has a graph theoretic answer.

The suppressed graph H of T [Gus88] is a bipartite mixed graph constructed as follows: (1) For each row or column of T, there is a unique node in H. (2) For each suppressed cell (i, j) of T, there is a unique edge in H between the node of row i and that of column j such that (a) if the

row column index	a	b	c	row sum
1	$\boxed{0}$	$\boxed{9}$	1	10
2	$\boxed{9}$	$\boxed{9}$	$\boxed{0}$	18
3	6	$\boxed{0}$	$\boxed{5}$	11
column sum	15	18	6	



In the 3×3 table, the number in each cell is the value of the cell. A cell with a box is a suppressed cell. The lower and upper bounds of the boxed cells are 0 and 9. The graph is the suppressed graph of the table. Node R_i corresponds to row i, and node C_j to column j.

Figure 2: The Suppressed Graph

value of the cell is strictly between its bounds, then the edge is undirected, (b) if the value is equal to the lower bound, then the edge points from the row node to the column node, and (c) if the value is equal to the upper bound, then the edge points from the column node to the row node. Figure 2 illustrates a table and its suppressed graph.

A *traversable* cycle in H is any cycle that can be traversed along the directions of its edges; a *direction-blind* cycle in H refers to an ordinary cycle disregarding directions of edges. The *direction-blind cycle space of* H, denoted by $CS(H)$, is the Z_2 vector space consisting of the mod 2 sums of direction-blind edge-simple cycles in H. In other words, the vector addition of two edge-simple cycles results in the set of edges that appear on exactly one of the two cycles. $CS(H)$ is a classic concept [Ber85]. The *traversable cycle space of* H, denoted by $TCS(H)$, is the Z_2 vector space consisting of the mod 2 sums of traversable edge-simple cycles in H. An element of $TCS(H)$ need not be a traversable edge-simple cycle or an edge-disjoint set of them. $TCS(H)$ is a subspace of $CS(H)$ and is a new concept.

The cycles and cycle spaces can be related to the feasible assignments and kernels through two edge labeling processes as follows. The first labeling process is the *direction-blind labeling*: because H is bipartite, every direction-blind edge-simple cycle of H is of even length; consequently, the edges on the cycle can be alternatingly labeled $+1$ and -1. Because every element in $CS(H)$ is a collection of pairwise edge-disjoint direction-blind edge-simple cycles, the labeling process can be extended to every element of $CS(H)$ by labeling each cycle of the element.

Lemma 2 *A direction-blindly labeled element of $CS(H)$ is an element of $UK(T)$.*

Proof. First of all, any labeled element of $CS(H)$ can be regarded as an assignment to the suppressed cells: if the corresponding edge of a suppressed cell is in the element, then the value assigned to the cell is the label of the edge; otherwise, the value is 0. This assignment need not satisfy the bounds of suppressed cells; however, as shown below, it has to preserve all row and column sums.

Let O be the assignment consisting of the original values of the suppressed cells in T; let A be any direction-blindly labeled element of $CS(H)$. To show $A \in UK(T)$, it suffices that $O + A$ and O have the same row and column sums, or in other words, the sum of A over the suppressed cells in any column or row R is 0. Because the row R in T is actually a node in H and because the suppressed cells of the row R in T are the edges incident to the node R in H, the sum of A over the suppressed cells of the row R in T equals the sum of A over the edges incident to the node R

in H; consequently, the alternating labeling rule guarantees both sums to be 0. ∎

The second labeling process is called the *traversable labeling* and applies to traversable cycles: the edges on any traversable edge-simple cycle of H can be alternatingly labeled $+1$ and -1 such that: (1) all directed edges from row nodes to column nodes are labeled $+1$, (2) all directed edges from column nodes to row nodes are labeled -1, and (3) the undirected edges may be labeled $+1$ or -1 subject to the alternating labeling rule. Furthermore, the labeling process can be extended to any $TCS(H)$ element decomposable into a collection of pairwise edge-disjoint traversable edge-simple cycles; however, not all elements in $TCS(H)$ have such decompositions.

Lemma 3 *A traversably labeled element of $TCS(H)$ is an element in $BK(T)$.*

Proof. Let O be the assignment consisting of the original values of the suppressed cells in T; let B be any traversably labeled element of $TCS(H)$. To show $B \in BK(T)$, it suffices to find a non-zero number s such that $O + s \cdot B$ is a bounded assignment of T. The choice of s will become obvious after the following explanation of the traversable labeling. In the traversable labeling, an edge is labeled $+1$ only if either it is an undirected edge or it is a directed edge from a row node to a column node. In either case, the value of the suppressed cell corresponding to the edge is less than the upper bound of the cell. Therefore to derive a new bounded feasible assignment, one may increase the value of the cell corresponding to the edge; similarly if an edge is labeled -1, one may decrease the value of its corresponding cell. Now s can be explicitly chosen as follows. Let $V_{i,j}$, $U_{i,j}$, and $L_{i,j}$ denote, respectively, the value, the upper bound, and the lower bound of any cell (i,j). Let $s = \min \{U_{i,j} - V_{i,j} \mid \text{Edge } (i,j) \text{ is labeled } +1 \text{ in } B.\} \cup \{V_{i,j} - L_{i,j} \mid \text{Edge } (i,j) \text{ is labeled } -1 \text{ in } B.\}$; when the minimum is $+\infty$, let s be an arbitrary positive number. The traversable labeling rules and the minimality of s imply that $O + s \cdot B$ satisfies the bounds of the suppressed cells; the alternating labeling rule guarantees that both $O + s \cdot B$ and O yield the same row and column sums. ∎

2.2 The Main Results

Lemmas 2 and 3 have given a basic relationship between the cycle spaces and the kernels at the level of *vectors.* Below, the Strongly Connected Basis Theorem will describe a deep relationship at the level of *vector bases.* To state the theorem, a simple extension of the labeling processes is needed for vector bases. A *direction-blindly labeled basis of* $CS(H)$ is defined to be a basis of $CS(H)$ such that all basis elements are direction-blindly labeled; a *traversably labeled basis of* $TCS(H)$ is defined to be a basis of $TCS(H)$ such that all basis elements are traversably labeled.

Theorem 4 (*The Strongly Connected Basis Theorem*)

Let T be any table and H be the suppressed graph of T. Further let $H_1, \ldots, H_k$ be the strongly connected components of H, and let B_i be an arbitrary direction-blindly labeled basis of $CS(H_i)$. Then $\cup_{i=1}^{k} B_i$ is a basis of $BK(T)$.

The Strongly Connected Basis Theorem will be proved in the next section; here the theorem is immediately applied to solve the Linear Invariant Test Problem.

Theorem 5 *Let $MM(n)$ be the sequential time for multiplying two $n \times n$ integer matrices in Strassen's fashion* [CW87] [AHU74]. *Then the Linear Invariant Test Problem can be solved in $O(n + e)$ sequential time and in $O(\log^2 n)$ parallel time using $MM(n)$ processors on a EREW PRAM for any input table of n rows and columns and e suppressed cells.*

Proof. Lemma 1 and the Strongly Connected Basis Theorem can be used to solve the Linear Invariant Test Problem as follows. First, compute the strongly connected components $H_1, \ldots, H_k$ of H. Second, find a direction-blindly labeled basis for each $CS(H_i)$ as follows. Find an *arbitrary* direction-blind spanning tree of H_i. Choose the cycles formed by each non-tree edge and the tree path between its two ends; call this cycle the *spanning cycle* of the non-tree edge. It is a classic fact that these spanning cycles form a basis of $CS(H_i)$ [Ber85]. Now direction-blindly label each spanning cycle. Finally, from Lemma 1 and the Strongly Connected Basis Theorem, f is an invariant if and only if the values of f at the labeled spanning cycles of each H_i are all zero; to compute these values, choose an arbitrary root for the spanning tree of H_i, and do the following:

1. For each tree edge, compute its depth; the tree edges incident to the root are of depth 1. Then compute its depth label: -1 for odd depth and $+1$ for even depth. Furthermore, compute its single-term value: the product of its depth label and its coefficient in f.

2. For each node, compute its path sum: the sum of the single-term values over the tree path from the root to the node.

3. For each non-tree edge, compute its cycle sum: the difference between the two path sums of its ends plus or minus (to be determined later) the coefficient of the edge in f. The cycle sum is to be the value of f at the spanning cycle for the non-tree edge; the plus-minus sign for the coefficient of the edge is chosen to direction-blindly label the cycle.

Sequential Complexity. The strongly connected components of H can be obtained in $O(n+e)$ time because H has n nodes and e edges. By performing undirected depth-first search in each strongly connected component, one can compute in linear time a spanning tree and its associated depths, depth labels, single-term values, path sums, and cycle sums. Therefore, the total time complexity is $O(n+e)$.

Parallel Complexity. The strongly connected components of H can be found in $O(\log^2 n)$ time and $MM(n)$ processors by computing transitive closure. For each strongly connected component, one can obtain a breadth-first search tree and its associated depths, depth labels, and single-term values in $O(\log n^2)$ time and $MM(n)$ processors using a bread-first search algorithm by Gazit and Miller [GM87]. Then the path sums can be computed in $O(\log^2 n)$ time and $O(n)$ processors using a doubling-up process. With the path sums, one can find the cycle sums and consequently the answer in $O(\log n)$ and $O(e)$ processors. Therefore, the total complexity is $O(\log^2 n)$ time and $MM(n)$ processors. ∎

3 The Proof of the Strongly Connected Basis Theorem

The Strongly Connected Basis Theorem will be proved in three stages. Subsection 3.1 examines the question of deciding whether $CS(H) = TCS(H)$, which is the graph theoretic counterpart of that of deciding whether $UK(T) = BK(T)$. Subsection 3.2 proves an unbounded version of the Strongly Connected Basis Theorem. Subsection 3.3 completes the proof and develops for subsequent papers another important theorem, namely, the *Bound Elimination Theorem.*

3.1 When Is CS(H) = TCS(H)?

Theorem 6 (*The CS-TCS Theorem*)

$TCS(H) = CS(H)$ if H is strongly connected.

Put another way, the CS-TCS theorem says that if H is strongly connected, then $CS(H)$ has a basis consisting of traversable cycles, and consequently any basis of $CS(H)$ is also a basis of $TCS(H)$ even if the basis has non-traversable cycles. The CS-TCS Theorem has two obviously equivalent variations stated as the following corollaries; all three versions hold for non-bipartite graphs as well as bipartite graphs.

Corollary 7 *$TCS(H) = CS(H)$ if every connected component of H is strongly connected.*

Corollary 8 *$TCS(H) = CS(H - O)$, where O is the set of edges in H but not in any strongly connected components of H.*

The proof of the CS-TCS Theorem uses mathematical induction on two orders derived from depth-first search, or DFS for short. A detailed discussion of DFS can be found in many textbooks such as [AHU74]; a brief discussion is given here. Because H is strongly connected, DFS produces a single spanning tree; because H may contain directed edges, there may be three types of non-tree edges, namely, back edges, forward edges, and cross edges. A back edge points from a node to an ancestor in the tree; a forward edge points from a node to a descendant; a cross edge is between two nodes of no ancestor-descendant relationship. During the depth-first search, the nodes are numbered in increasing order according to the sequence in which the nodes are last visited, that is, a node x is given its number $dfs(x)$ when the search backs up from x. For induction purpose, for any directed non-tree edge $e = x \rightarrow y$, let $dfs(e)$ denote $dfs(x)$. Furthermore, for ease of discussion, assume that the nodes are placed in such a way that for any two nodes x and y, if $x <_{dfs} y$, then y is either above x in the DFS spanning tree or to the right of x. Hence, all cross edges point from right to left.

The two orders, both denoted by $<_{dfs}$, are defined as follows: (1) For any nodes x and y, $x <_{dfs} y$ if and only if $dfs(x) < dfs(y)$. (2) For any directed non-tree edges d and e, $d <_{dfs} e$ if and only if $dfs(d) < dfs(e)$. The next two technical lemmas are developed for induction on the orders.

Lemma 9 *For every directed non-tree edge $e = x \rightarrow y$, there is a traversable edge-simple path P from y to some node z such that (1) z either is equal to x itself or is above x in the DFS spanning tree, and (2) all directed non-tree edges in P are $<_{dfs}$ e.*

Proof. P and z are built as follows. Because H is strongly connected, there is a traversable edge-simple path $Q = u_1, ..., u_k$ from y to x. Let s be the smallest index in Q such that $x \leq_{dfs} u_s$;

s exists because $u_k = x$. Now let $P = u_1, ..., u_s$ and let $z = u_s$. The two properties of P and z are verified as follows. Property (1): Let R be the traversable path formed by e and P; in other words, $R = u_0, u_1, ..., u_s$ where $u_0 = x$. From the choice of s, u_{s-1} is either equal to x, or below x, or to the left of x. If $x \neq u_s$, then $x <_{dfs} u_s$ and consequently u_s is either above x or to the right of x. However, the latter case cannot happen. Otherwise, u_{s-1} and u_s would form either an undirected cross edge or a directed cross edge pointing from left to right. Neither case can happen in DFS. Property (2): Because P is traversable, any directed non-tree edge in P points from some node u_i with $i < s$. The minimality of s implies $u_i <_{dfs} x$. Hence the edge is $<_{dfs} e$. ■

Lemma 10 *For every directed non-tree edge e, there is a traversable edge-simple cycle E containing e such that all directed non-tree edges in E except e itself are $<_{dfs} e$.*

Proof. Let $e = x \to y$; let path P and node z be as described in Lemma 9. From Property (1) of P and z, the DFS tree has a traversable edge-simple path L from z to x. Now let E be the cycle formed by e, P and L. Then C contains e and is traversable edge-simple, Moreover, Property (2) of P and z guarantees that all non-tree edges in E except e itself be $<_{dfs} e$ because L is a tree path. ■

Proof of the CS-TCS Theorem. Because $TCS(H)$ is a vector subspace of $CS(H)$, it suffices to find a basis B of $CS(H)$ and a set C of traversable cycles in H such that every element in B is a mod 2 sum of cycles in C.

Construct B as follows. For each non-tree edge e, let $S(e)$ be the direction-blind edge-simple cycle formed by e and the tree path between the ends of e in the DFS spanning tree; call $S(e)$ the *spanning cycle* of e. Let B be the set of $S(e)$'s; it is a classic result that B is a basis of $CS(H)$ [Ber85]. Build C as follows. For each non-tree edge e, a traversable edge-simple cycle $E(e)$ containing e is chosen according to whether e is directed or not. If e is undirected, let $E(e) =$

$S(e)$, which is traversable because every undirected non-tree edge is a back edge. If e is directed, let $E(e)$ be the cycle described in Lemma 10.

Decompose $S(e)$ for induction purpose as follows. Let $u_1, \ldots, u_s$ be the undirected non-tree edges in $E(e)$ other than e itself; also let $d_1, \ldots, d_t$ be the directed non-tree edges in $E(e)$ other than e itself. Then for any non-tree edge e, $0 = S(e) + E(e) + \sum_{i=1}^{s} \mathrm{S}(u_i) + \sum_{j=1}^{t} \mathrm{S}(d_j)$ where $+$ is mod 2. This is true because (1) the RHS sum is in $CS(H)$, and (2) in the sum each non-tree edge of $E(e)$ is canceled for appearing once in E(e) and once in the spanning cycles. In other words, the sum contains only tree edges, and hence it must be the zero cycle. Now rewrite the equality as $S(e) = E(e) + \sum_{i=1}^{s} S(u_i) + \sum_{j=1}^{t} S(d_j)$, called the *traversable decomposition* of $S(e)$.

Now it can be shown by induction that every $S(e)$ in B is a sum of cycles in C. There are two cases based on whether e is directed or not. If e is undirected, $S(e) = E(e)$ is already a member of C. If e is directed, apply induction on $<_{dfs}$ for the directed non-tree edges. Induction Hypothesis: For each directed non-tree edge $g <_{dfs} e$, $S(g)$ is a sum of cycles in C. Induction Step: In the RHS of $S(e)$'s traversable decomposition, $S(u_i) = E(u_i)$ because u_i is undirected. Furthermore, the induction hypothesis implies that $S(d_i)$ is a sum of cycles in C because on $E(e)$ every directed non-tree edge except e itself is $<_{dfs} e$. Therefore $S(e)$ is a sum of cycles in C. ∎

3.2 The Unbounded Version of the Strongly Connected Basis Theorem

Theorem 11 (*The CS-UK Theorem*)

Every direction-blindly labeled basis of $CS(H)$ is a basis of $UK(T)$; consequently, $\dim CS(H) = \dim UK(T)$.

Proof. From Lemma 2, any direction-blindly labeled basis of $CS(H)$ is a subset of $UK(T)$. Therefore, it suffices to show that (1) every direction-blindly labeled basis of $CS(H)$ is linearly independent in $UK(T)$, and (2) $CS(H)$ has some direction-blindly labeled basis B such that every element of $UK(T)$ is a linear combination of B. The two properties imply the dimension equality. The dimension equality and the first property in turn imply the basis embedding. These two properties are shown in the following two lemmas. ∎

Lemma 12 *Every direction-blindly labeled basis of $CS(H)$ is linearly independent in $UK(T)$.*

Proof. Let U be any direction-blindly labeled basis of $CS(H)$; let V be the unlabeled version of U; let k be the number of elements in U or V. Construct an integer matrix for U and a Z_2 matrix for V as follows. The elements of U can be regarded as integer vectors consisting of $+1$, 0, or -1 components; let M_U be the integer matrix consisting of the elements of U as rows. Similarly, the elements of V are regarded as Z_2 vectors; let M_V be the Z_2 matrix consisting of the elements of V as rows. Then $M_U = M_V \pmod 2$. Because V is a basis of $CS(H)$, M_V contains a $k \times k$ submatrix S_V such that det $S_V \neq 0 \pmod 2$. Let S_U be the submatrix of M_U corresponding to S_V. Then $S_U = S_V \pmod 2$; consequently, $\det S_U = \det S_V \neq 0 \pmod 2$. This implies det S_U is a non-zero integer. Therefore, R is linearly independent in $UK(T)$. ∎.

Lemma 13 *$CS(H)$ has some direction-blindly labeled basis B such that every element of $UK(T)$ is a linear combination of B.*

Proof. First construct B as follows. Arbitrarily choose a spanning tree in each connected component of H; let $e_1, \ldots, e_k$ be the non-tree edges. For each e_i, there is a unique node-simple cycle $S(e_i)$ formed by e_i and the tree path between the ends of e_i; $S(e_i)$ is the so-called spanning cycle of edge e_i. Let B be the set of $S(e_i)$'s; B is known to be a basis of $CS(H)$ [Ber85]. Now direction-blindly label B such that w.l.o.g, each e_i is labeled $+1$ in its spanning cycle $S(e_i)$. The requirement for B is verified as follows. Let g be any element in $UK(T)$. Let g_i be the component of g at e_i. Let

$h = g - g_1 \cdot S(e_1) - \cdots - g_k \cdot S(e_k)$. To prove g is a linear combination of B, it suffices to show $h = 0$ as follows. On one hand, e_i appears in $S(e_i)$ but not in any other spanning cycle; moreover, in the RHS the components of e_i at g and $g_i \cdot S(e_i)$ cancel each other. Therefore, the components of h at the e_i's are all 0. On the other hand, Lemma 2 guarantees that all $S(e_i)$'s are in $UK(T)$. Hence h is also in $UK(T)$. In summary, h has the following two properties: (1) the components at the non-tree edges are all 0, and (2) for each node, the sum of the components at the edges incident to the node is 0. These two properties imply that the components of h are all 0. Therefore $h = 0$. ∎

3.3 The Proof of the Strongly Connected Basis Theorem

Lemma 14 *Every traversably labeled basis of $TCS(H)$ is linearly independent in $BK(T)$; consequently* $\dim TCS(H) \leq \dim BK(T)$.

Proof. Almost the same as Lemma 12. ∎

Remark: The proof of Lemma 13 cannot be extended to the case of $TCS(H)$ and $BK(T)$ because it is difficult to find a basis for $TCS(H)$ consisting of traversable cycles as regular as spanning cycles.

Theorem 15 (*The Bound Elimination Theorem*)

$UK(T) = BK(T)$ if every connected component of H is strongly connected.

Proof. Because $BK(T)$ is a vector subspace of $UK(T)$, it suffices that $\dim UK(T) \leq \dim BK(T)$, which is implied by the following dimension relationships: $\dim UK(T) = \dim CS(H)$ (the CS-UK Theorem), $\dim CS(H) = \dim TCS(H)$ (the CS-TCS Theorem), and $\dim TCS(H) \leq \dim BK(T)$ (Lemma 14). ∎

Theorem 16 (*Gusfield [Gus88]*) *A suppressed cell of T is not an invariant cell if and only if it is contained in an traversable edge-simple cycle of H.*

In the following, let O be the set of edges in H but not in its strongly connected components; also let W be the same table as T except that the cells corresponding to the edges in O are all published. Then the suppressed graph of W is $H - O$.

Lemma 17 $BK(T) = UK(W)$.

Proof. From Theorem 16, the components of any element in $BK(T)$ are always 0 at O. This means that every element of $BK(T)$ is in effect a member of $UK(W)$; consequently, $BK(T)$ is a vector subspace of $UK(W)$. So it suffices that dim $UK(W) \leq$ dim $BK(T)$, which is shown by the following dimension relationships: $\dim UK(W) = \dim CS(H - O)$ (the CS-UK Theorem), $\dim CS(H-O) = \dim TCS(H)$ (the CS-TCS Theorem), and $\dim TCS(H) \leq \dim BK(T)$ (Lemma 14). ■

Theorem 18 (*The TCS-BK Theorem*)

Every direction-blindly labeled basis of $TCS(H)$ is a basis of $BK(T)$; consequently, every direction-blindly labeled element of $TCS(H)$ is an element of $BK(T)$.

Proof. First, every direction-blindly labeled basis of $TCS(H)$ is also a direction-blindly labeled basis of CS($H - O$) from the CS-TCS Theorem. Second, every direction-blindly labeled basis of CS($H - O$) is a basis of $UK(W)$ from the CS-UK Theorem. Finally, every basis of $UK(W)$ is also a basis of $BK(T)$ from Lemma 17. ■

One can now use the above discussion to prove the Strongly Connected Basis Theorem in two steps: (1) because the strongly connected components of H are the connected components of $H - O$, from the CS-TCS Theorem $\cup_{i=1}^{k} B_i$ is also a direction-blindly labeled basis of TCS(H), and (2) from Theorem 18, the direction-blindly labeled basis is a basis of $BK(T)$.

4 Conclusions

For detecting and protecting information in two dimensional tables, this paper has developed a framework based on linear algebra and graph theory as opposed to linear programming. In this framework, the paper has obtained the Strongly Connected Basis Theorem; this theorem immediately yields a sequential linear-time algorithm and an NC algorithm for the Linear Invariant Test Problem. In contrast, the conventional approach for solving the problem is to use linear programming techniques; they take more than linear sequential time and are unlikely to be in NC. This paper has also developed the Bound Elimination Theorem. Together with the Strongly Connected Basis Theorem, they will be essential in further papers in preparation on other aspects of detecting and protecting information in tables.

5 Acknowledgements

The first author is grateful to Mike Fischer for providing financial support and other help at Yale University. The authors both would like to thank Larry Cox and Gordan Sande for helpful discussions and pointers.

References

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[BCS83] G.J. Brackstone, L. Chapman, and G. Sande. Protecting the confidentiality of individual statistical records in Canada. In *the Proceedings of the Conference of the European Statisticians 31st Plenary Session, Geneva*, 1983.

[Ber85] C. Berge. *Graphs.* North-Holland, New York, second revised edition, 1985.

[Coo85] S.A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[Cox75] L. Cox. Disclosure analysis and cell suppression. In *the Proceedings of the American Statistical Association, Social Statistics Section*, pages 380–382, 1975.

[Cox77] L. Cox. Suppression methodology in statistics disclosure. In *the Proceedings of the American Statistical Association, Social Statistics Section*, pages 750–755, 1977.

[Cox78] L. Cox. Automated statistical disclosure control. In *the Proceedings of the American Statistical Association, Survey Research Method Section*, pages 177–182, 1978.

[Cox80] L. Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association, Theory and Method Section*, 75(370), June 1980.

[CS79] L.H. Cox and G. Sande. Techniques for preserving statistical confidentiality. In *the Proceedings of the 42nd Session of the International Statistical Institute.* the International Association of Survey Statisticians, December 1979.

[CW87] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *the Proceedings of ACM Symposium on Theory of Computing*, pages 1–6, 1987.

[Den82] D. Denning. *Cryptography and Data Security.* Addison-Wesley, 1982.

[DLR79] D. Dobkin, R.J. Lipton, and S. Reiss. Linear programming is log-space hard for P. *Information Processing Letters*, 8:96–97, 1979.

[GM87] H. Gazit and G.L. Miller. A parallel algorithm for breadth-first search of a directed graph, 1987. manuscript.

[GSS82] L.M. Goldschlager, R.A. Shaw, and J. Staples. The maximum flow problem is log space complete for P. *Theoretical Computer Science*, 21:105–111, 1982.

[Gus87] D. Gusfield. Optimal mixed graph augmentation. *SIAM Journal on Computing*, 16(4):599–612, August 1987.

[Gus88] D. Gusfield. A graph theoretic approach to statistical data security. *SIAM Journal on Computing*, 17(3):552–571, June 1988.

[Kao86] M.Y. Kao. *Systematic Protection of Precise Information on Two Dimensional Cross Tabulated Tables.* PhD thesis, Yale University, 1986.

[Mur76] K. Murty. *Linear and Combinatorial Programming.* Wiley, 1976.

[PS82] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity.* Prentice-Hall, 1982.

[San77] G. Sande. Towards automated disclosure analysis for establishment based statistics. Technical report, Statistics Canada, December 1977.

[San78] G. Sande. A theorem concerning elementary aggregations in simple tables. Technical report, Statistics Canada, April 1978.

[San84] G. Sande. Automated cell suppression to preserve confidentiality of business statistics. *Statistical Journal of the United Nations*, 2:33–41, 1984.

[Sante] G. Sande. Confidentiality and polyhedra, an analysis of suppressed entries on cross tabulations. Technical report, Statistics Canada, unknown date.

[Vai87] P.M. Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations. In *the Proceedings of ACM Symposium on Theory of Computing*, pages 29–38, 1987.