This work was partially funded by the National Science Foundation under grant number MCS-81/05894

A GRAPH THEORETIC APPROACH TO STATISTICAL DATA SECURITY

Dan Gusfield
YALEU/DCS/TR-326
August, 1984

A GRAPH THEORETIC APPROACH TO STATISTICAL DATA SECURITY

Dan Gusfield
Department of Computer Science
Yale University
August 1984

Abstract

We study the problem of protecting sensitive data in a two dimensional table of integer statistics when non-sensitive data and row and column sums are made public. We give a linear time algorithm which, given the complete table, identifies sensitive data that is exactly determinable from the non-sensitive data; we give a fast algorithm which, given only the non-sensitive data, identifies sensitive data that is exactly determinable from the non-sensitive data; for the case that all data are positive, we give a fast algorithm to determine the fewest number of additional non-sensitive data cells to withhold in order that no withheld data is determinable from the released data; we discuss the same problem in the case that the sensitive data is positive, but some non-sensitive data can be zero; we show how to use simple (no costs) network flow to compute the tightest upper and lower bounds on any withheld data, given the released data; we discuss computing tightest upper and lower bounds in the special case that only the row and column sums are released - we show that trivial methods suffice in that case, and in fact, that the lower bounds can be computed with O(n+m) arithmetic and comparison operations in an nxm table.

This work was partially funded by the National Science Foundation under grant numbers MCS-81/05894.

Table of Contents

1. Introduction

In this paper we study the problem of protecting sensitive data in a two dimensional table of statistics when the non-sensitive data is made public. Work in this area was begun by Cox and Sande [COX75], [COX77], [COX78], [COX80] and reported in Denning [DEN]. The general problem is motivated by concerns for privacy and security. For a more complete discussion of background and motivation see [DEN].

1.1. Problem Statements, Definitions and Main Results

The basic setting for the paper is that one party (the census bureau, say) has a two-dimensional table, D, of cross tabulated integer statistics; each entry D(i,j) is a non-negative integer in cell (i,j) of D; R(i) is the sum of the cell values in the i'th row of D, and C(j) is the sum of the cell values in the j'th column of D. We assume that D has at least two rows and at least two columns, and that each R(i) and C(j) is strictly positive. All the row and column sums are to be made public (disclosed) along with *some* of the cell values, but the remaining cell values, called *sensitive* values, are to be *suppressed* (not disclosed). Unless care is taken, however, the disclosure of the non-sensitive values might allow an adversary to deduce the *exact* value of one or more of the sensitive values. Hence to avoid disclosing the exact value of any sensitive cell, the values of some non-sensitive cells may also need to be suppressed; these suppressions are called *complementary suppressions*. In a table D, the set of *suppressed cells* consists of the sensitive cells along with any complementary suppressed cells.

Given a table with suppressions, a suppressed cell is called *protected* if its exact value is not determinable from knowledge of only the row and column totals, and the values of the unsuppressed cells. To make this precise, let X be the set of all suppressed cells and for cell (i,j) $\in$ X, let x(i,j) be a variable denoting the value of the cell. For each row i and column j, let R*(i) and C*(j) be respectively the sum of the undisclosed values in row i of D, and the sum of the

undisclosed values in column j of D.

Definition 1.1: We define a *legal solution* of D as a non-negative integer solution to the following system of linear equalities:

For each fixed i, $\sum_{j \text{ such that } (i,j) \in X} x(i,j) = R^*(i)$

and

for each fixed j, $\sum_{i \text{ such that } (i,j) \in X} x(i,j) = C^*(j)$.

Definition 1.2: A cell (i,j) in X is *protected* if and only if there exists two legal solutions x_1 and x_2, where $x_1(i,j) \neq x_2(i,j)$.

Definition 1.3: A table with suppressions is protected if and only if all of its suppressed cells are protected.

Notice that this definition of protection essentially implies that side information such as correlations between cell values or special bounds on cell values etc. is unknown to the adversary, or at least does not restrict the set of legal solutions. The results in this paper can easily be extended to the cases when given upper and lower bounds on the values of individual cells are known to the adversary (provided that the census bureau knows what bounds the adversary is assuming). Notice also that the assumption that each R(i) and C(j) is strictly positive is reasonable, since there is no way to protect the (all zero) entries in such a row or column.

In this paper we discuss four problems:

a) (Census Bureau problem) Given the *complete table*, D, and a set of cells, X, determine whether table D with cells X suppressed is protected. We give an algorithm that identifies all unprotected cells and runs in linear time in the size of X.

problem b) by trial and error, iteratively using the upper and lower bounds computed by c), and problem a) also by computing upper and lower bounds and checking for equality. Hence this paper provides substantial theoretical and practical improvements for problems a), b) and d), and a moderate theoretical but perhaps not practical improvement in problem c).

We will discuss the four problems in order, but first present a graphical framework that will be used throughout the paper.

2. A Network Flow Framework

It will be useful to represent the table D as a directed bipartite graph, H, and a legal solution of D as a flow in the related network G. We use the notation <i,j> to refer to a directed edge from node i to node j, and we use the notation (i,j) to refer to an undirected edge connecting nodes i and j.

Definition 2.1: Graph H is a directed bipartite graph with node set R on one side representing the rows of D, and node set C on the other side representing the columns of D. A directed edge extends from node i on R to node j on C if and only if cell (i,j) is suppressed in D.

Definition 2.2: Network G is constructed from graph H as follows: add a node s to H and a directed edge from s to each node of R, and add a node t to H and a directed edge from each node of C to t. In G, the capacity of each edge <s,i> is set to $R^*(i)$; the capacity of each edge <j,t> is set to $C^*(j)$; and the capacity of every edge in H is set to infinity. Let T be the sum of the capacities of the edges out of s (T is also the sum of the edge capacities of the edges into t).

Definition 2.3: A flow function F is an assignment of a non-negative real number, F(i,j), to each directed edge <i,j> of G such that F(i,j) is less than or equal to the capacity of edge <i,j> and so that for every node $i \neq s,t$ $\Sigma F(i,j) = \Sigma F(j,i)$. The quantity Σ F(s,i) is called the *value* of the flow, and it equals Σ F(i,t). A flow is called a *maximum flow* if its value is

maximum over all flows in G. A flow is called *integral* if F(i,j) is an integer for every edge <i,j>.

The example in Figure 1 illustrates the above definitions.

It is known [FF] that a maximum integral flow exists if the capacities on all the edges are integers. The following fact is immediate.

Fact 2.1: Let D and G be as above and let x be any legal solution of D. Then the assignment F(i,j) <-- x(i,j) is an integral maximum flow in G of value T, i.e. F is an integral flow that saturates all edges out of s, and these edges form an s-t cut in G. Conversely, any integral maximum flow in G (of value T) defines a solution to D in the analogous way: x(i,j) <-- F(i,j). Hence a suppressed cell (i,j) in D is protected if and only if there exists two integral flows F_1 and F_2, each of value T, such that $F_1(i,j) \neq F_2(i,j)$.

In the following sections we will use this fact in the solution to problems a), a'), c) and d), and in our approach to problem b).

Definition 2.4: For an edge <u,v> in G let C(u,v) denote the (upper) capacity of edge <u,v> and let L(u,v) denote its lower capacity.

In definition 2.2, C(s,i) and C(j,t) are respectively R*(i) and C*(j), $C(i,j) = \infty$ for <i,j> in H, and L(u,v) = 0 for each edge <u,v> in G.

To demonstrate the utility of this framework, we recast problem c) in terms of network flow. If we think of C(i,j) and L(i,j) as variable parameters, then the best lower bound on the value of cell (i,j) is the minimum value that C(i,j) can take on (while all other edges in G have their original capacities as given in definition 2.2) so that there is a flow of value T; similarly, the best upper bound is the maximum value L(i,j) can take on so that there is a flow of value T. In

section 5 we will solve each of these parametric problems using a single network flow computation.

3. Determining if D is protected

In this section we give a linear time algorithm to determine if D is protected.

Definition 3.1: Let F be a flow of value T in G, and let F* be the following graph derived from H and F: F* is a directed graph on the same node set as H; for any pair $i \in R$ and $j \in C$, F* contains a directed edge $<i,j>$ of infinite capacity, and F* contains a directed edge $<j,i>$ of capacity F(i,j) provided that $F(i,j) > 0$.

Given a particular flow F in G, a suppressed cell (i,j) in D is clearly protected if either edge $<i,j>$ or $<j,i>$ (if $<j,i>$ exists) is in a simple directed cycle SC of length four or more in F*. In that case, simply augment one or more units of flow around SC, obtaining another flow of value T in which F(i,j) is integral but different from its previous value: if $<i,j>$ is on SC then F(i,j) increases and if $<j,i>$ is on SC, then F(i,j) decreases. Notice that a cycle of length two in F* merely implies that $F(i,j) > 0$, and augmentation around that cycle has no affect on the flow.

On the other hand, what if neither of the edges $<i,j>$ or $<j,i>$ are in an appropriate cycle in a particular F*? Does it automatically follow that cell (i,j) is unprotected?

Theorem 1: Let D and G be as above, and let F be *any* integral maximum flow in G, with the associated derived graph F*. Then a suppressed cell (i,j) in D is protected if and only if at least one of the edges $<i,j>$ or $<j,i>$ is contained in a simple directed cycle of length four or more in F*.

Proof: The sufficiency was noted above. To prove necessity, suppose cell (i,j) is protected and let F′ be another maximum integral flow in G such that $F(i,j) \neq F'(i,j)$. Define FF(i,j) =

F'(i,j) - F(i,j), for each <i,j> in H. We represent FF as a weighted bipartite graph, FF*, containing directed edge <i,j> with weight FF(i,j), if FF(i,j) > 0, and containing edge <j,i> with weight FF(i,j), if FF(i,j) < 0. It is easy to see that FF* contains at most one edge of either <i,j> or <j,i>, and that FF* is a subgraph of F*. We will suppose that FF* contains edge <i,j> and show that <i,j> is contained in a simple directed cycle (of length four or more) in FF*, and hence <i,j> is in the same cycle in F*.

First, note that for any fixed node i in FF*, the weights of all the edges incident with i sum to zero. Hence any node i in FF* that is incident with at least one edge is incident with at least one edge directed into i and one directed out of i; we call this the *balance* property. It follows easily from this fact that there exists a directed cycle, SC, in FF*. Since FF* is bipartite and never contains both edges <i,j> and <j,i>, SC must contain a simple cycle of length four or more. We assume, wlog, that SC itself is such a cycle. Let f be the minimum absolute value of the weights of the edges on SC. We subtract f from each positive weight on SC, add f to each negative weight on SC, and remove any edges with resulting weight zero (there is at least one); the result is another subgraph of F* which again has the balance property. We can therefore continue finding directed cycles in FF*, changing weights and removing edges each time, until there are no more edges left. Edge <i,j> must be in at least one of these cycles. □

We will now transform F* into more convenient form. Suppose F(i,j) > 0 so that both <i,j> and <j.i> appear in F*. By Theorem 1, cell (i,j) is protected if and only if *at least one* of these edges are in an appropriate directed cycle. This fact permits the convenient transformation of F* from a directed graph into a *mixed* graph (one with both directed and undirected edges): replace every pair of directed edges <i,j> and <j,i> in F* by a single undirected edge (i,j). We assume from here on that F* is such a mixed graph. Notice that in the transformed F* there are now no cycles of length less than four, hence any edge on any cycle in F* is on a simple cycle of

length at least four. For the purposes of determining if D is protected, we can also ignore the actual capacities on F*, since it is clear from Theorem 1 that only the directions of the edges in F* matter. This means that the only distinction that matters (in determining protection) is whether D(i,j) is zero or not. Notice that the question of protection is also answered without needing to explicitly know R*(i) or C*(j). Figure 2 shows F* for F given in figure 1b.

Definition 3.2: Where the distinction does not matter, we will use the term "edge" and the notation (i,j) to refer to either the undirected edge (i,j) or the directed edge <i,j>.

Definition 3.3: A cycle in a mixed graph is called *traversable* if it is possible to walk around it without ever going opposite to the direction of any of the directed edges on it. An edge is called traversable if and only if it is contained on some traversable cycle.

In this terminology Theorem 1 becomes

Theorem 1′: Let F be *any* integral maximum flow in G, and let F* be the associated derived graph. A suppressed cell (i,j) is protected if and only if edge (i,j) in F* is traversable.

3.1. Graph CG

By theorem 1′ problem a) can be solved by determining which (if any) of the edges of F* are in no traversable cycles in F*. We solve this problem with a linear time algorithm that constructs the mixed graph CG; the edges of CG are exactly those edges of F* that are in no traversable cycles in CG. Graph CG will also be central in solving the problem of minimizing the number of complementary suppressions. The construction of CG uses the graph theoretic objects of *bridges* and *strong components*. Definitions of these objects and algorithms to find them are fully developed in a number of texts, such as [AHU] and [EVE].

Construction of graph CG

1. Find the strongly connected components of F* (undirected edges in F* are treated as two oppositely directed edges). Let A be the set of edges running between these components. Clearly all the edges of A are directed edges in F*.

2. Consider each of the strong components as a separate undirected graph, ignoring the directions on any directed edges. Find all (if any) bridges in each of the separate graphs, and let B be the set of bridges found. It is easily proved that any edge in B is an undirected edge in F*.

3. Let K be the graph induced by the edges of F* minus A ∪ B, and let K(1),..., K(r) be the connected components of K. Graph CG is formed by condensing each K(i) in F*. Hence each node in CG is in 1-1 correspondence with some K(i), and the edges of CG are exactly A ∪ B.

In Figure 3, graph CG is constructed from F* shown in Figure 2.

Theorem 2: Edge (i,j) is in no traversable cycle in F* if and only if (i,j) is in A ∪ B. That is, cell (i,j) is unprotected if and only if nodes i and j are in different components of K.

Proof: First any edge which is in a traversable cycle in F* is in some strong component of F* , and so no edge in set A is in a traversable cycle. Second, any directed edge <u,v> in a strong component is traversable, since, by definition of strong component, there must be a traversable path P from v to u in F*. So edge <u,v> followed by path P is a traversable cycle in F*. Hence A exactly contains the directed edges that are on no traversable cycles. Now we consider which undirected edges of F* are traversable. Since no edge in A is in a traversable cycle, we can delete set A from F* without affecting which undirected edges are traversable. Any bridges in the resulting graph are clearly non-traversable, so all edges in B are non-traversable. What remains to be shown is that any undirected edge not in B is in some traversable cycle in F*. Since (u,v) is not a bridge in the strong component it is contained in, there must be a path P in

F^* connecting (ignoring the directions of the edges on P) node u and node v. If P is traversable in either direction, we are done. If P is not traversable, let $<x,y>$ be the first directed edge on P such that the walk from u to v on P goes in the wrong direction, i.e. from y to x. Now $<x,y>$ is not in set A and so it is on some traversable cycle in F^*, and so there is a traversable path, P′, from y to x in F^*. Now the path consisting of P to y, P′ to x, and P to v is a path from u to v which traverses in the wrong direction one less edge than does P. Hence by repeating this argument, a traversable path in F^* from u to v can be found, and hence (u,v) is in a traversable cycle in F^*. □

Hence problem a) can be solved in time linear in the size of X if we use the flow F determined by D, since then CG can be built from D in time linear in the size of X. It is well known [AHU] [EVE] that strong components and bridges can be found in time linear in the number of edges in the graph.

Figure 3 shows that cells (1,4), (3,2), (3,3) and (3,5) of table D (Figure 1) are unprotected.

3.2. Solving the adversary problem a′

Theorem 3: Let F_1 and F_2 be any two maximum integral flows in G, and let CG_1 and CG_2 be the associated condensed graphs as defined above. Then $CG_1 = CG_2$.

Proof: This follows by combining Theorems 1 and 2. □

Hence the condensed graph CG is a function of G alone and not of F^*. This leads to the solution of problem a′. Given table D with suppressions the adversary forms G, computes any maximum integral flow F in G and then forms CG. Any cells associated with edges in sets A or B are unprotected: the cells of set A have exact value zero and those of set B have values given by F. Note that problem a) is solved in linear time while problem a′ first requires the computation of a network flow. These two solutions are essentially the same, but in problem a),

the census bureau has the advantage that it knows a particular maximum flow for free, namely the one corresponding to D itself.

4. Optimizing Complementary Suppressions

In this section, we will use F^* and CG to represent and help determine the optimal complementary suppressions in D; we will see that this problem can be cast as a graph augmentation problem on CG.

Let F be any integral maximum flow in G and let F^* be the associated derived graph. Each complementary suppression in D changes G, F and F^*: if cell (i,j) is suppressed then the directed edge <i,j> is added to G, the capacities of edges <s,i> and <j,t> in G are both increased by D(i,j), and a new maximum integral flow is obtained by setting F(i,j) equal to D(i,j) and leaving all the other flow assignments as they are in the previous flow. Hence, if D(i,j) is zero then edge <i,j> is added to F^*, and if $D(i,j) > 0$ then the undirected edge (i,j) is added to F^*. This direct correspondence between cell suppressions in D and edge additions in F^*, combined with Theorem 1′, implies that the complementary suppression problem can be solved by taking any F^* and adding the fewest number of edges corresponding to complementary suppressions in D, so that every edge in the resulting graph is in some traversable cycle. Problems of this type are called *graph augmentation* problems.

It is easy to see (and prove) that the complementary suppression problem can be solved using CG in place of F^*, i.e. the smallest set of edges needed to add to CG so that every edge is traversable is optimal for F^* and visa versa. What makes this augmentation problem difficult is that only those edges corresponding to unsuppressed cells in D can be added to CG, and if (i,j) is an unsuppressed cell where $D(i,j) = 0$, then the edge corresponding to cell (i,j) is the *directed* edge <i,j>.

In general we do not know the status of the above graph augmentation problem, but CG is still useful, since one can see quickly the effect of any proposed complementary suppressions. This leads to efficient implementation of heuristics (such as those suggested in [COX80]) based on hill climbing and incremental optimization, i.e. incrementally selecting a suppression that makes the most edges traversable, updating CG, and repeating until every edge is in some traversable cycle.

Although we don't know a solution to the general complementary suppression problem, there are useful special cases that can be efficiently solved. In the following section, we solve the case when D(i,j) > 0 for every cell in D, and then apply this to the case when D(i,j) > 0 for every sensitive cell, but where other cells in D may have zero value.

4.1. Special Case: D Strictly Positive

In this section we optimally and efficiently solve the complimentary suppression problem under the assumption that D(i,j) > 0 for every cell in D. We present a simple $O(|X|^2)$ time method based on a related problem and solution studied by Eswaren and Tarjan [ET]. In [G] we examine a more general class of graph augmentation problems (containing the special case discussed here) and give a solution to those problems that solves the special case in O(|X|) time. The faster solution is more involved than the one given here.

4.1.1. Unlabelled graph augmentation

In [ET] the following problem is solved: Given a forest, FG, of undirected trees, add the fewest number of undirected edges to FG so that every edge is in a simple cycle (hence of length at least three), where it is permitted to add an edge between any pair of nodes in FG. In our applications, we will only need the solution to this problem when FG is a single tree and contains at least three leaves (isolated nodes never arise in our applications, and the complementary suppression problem for two leaves is easy to solve directly). We sketch the solution of the

unlabelled graph augmentation problem given in [ET], for the case of FG a single tree.

Algorithm ET

1. Number the leaves of CG in pre-order, i.e. traversing the tree by depth-first search, the leaves are numbered in the order that they are reached. Let v(i) be the i'th leaf reached.

2. Suppose FG has p leaves and $k = \lfloor p/2 \rfloor$ Add an edge between leaves v(i) and v(k+i), for i = 1 through k. If p is odd (p = 2k+1), then add an edge between v(p) and any other leaf in FG.

For a proof that this solution is correct, see [ET]. Note that algorithm ET clearly runs in time linear in the number of edges of FG.

4.1.2. Labelled augmentation: modifying ET

When table D is strictly positive, CG is an forest of undirected trees. If we take CG in place of FG and use algorithm ET (adding $\lceil p/2 \rceil$ edges and yielding graph CG*) we may not end up with a solution to the complimentary suppression problem since "illegal" edges (those not corresponding to unsuppressed cells in D) may have been added. We give here an $O(|X|^2)$ time algorithm that takes CG* and transforms it to obtain an optimal solution to the suppression problem. We first need some definitions and observations.

Definition 4.1: In CG we give a node the label of R if it is a node on the R side of G, we give it the label C if it is on the C side of G, and we give it the label B if it is a condensed node and therefore contains nodes of G from both R and C. Let r, c and b be the number of leaves of CG labelled R, C and B respectively.

Given this notation, the complementary suppression problem for the case that D is strictly positive is a graph augmentation problem on CG with the constraint that no (R,R) edges or (C,C) edges may be added. We will refer to this graph augmentation problem as the *labelled augmentation problem.*

Fact 4.1: Assuming $r \leq c$, a simple lower bound on the size of the optimal solution of the labelled augmentation problem is c if $r + b \leq c$, and is $\lceil (r + c + b)/2 \rceil$ otherwise. We will see that this bound can be exactly met.

Assumption: We will assume that CG has $p = r+b+c > 2$ leaves, that p is even, that $r \leq c$, and until section 4.1.3, that $r+b \geq c$. The complimentary suppression problem is trivial when $p = 2$. When p is odd, we reduce the analysis to the even case as follows: If $r+b \geq c$ and p odd, it follows that $r+b > c$, and we remove from CG a B leaf and the unique path from it to a node of degree three of more, creating a forest with an even number of leaves with $r+b \geq c$. The solution to the labelled augmentation problem when p is odd is then obtained by taking the solution to the reduced case (p even) and adding an edge from the deleted B leaf to any other leaf. In section 4.1.3 we will briefly discuss the case that $r+b < c$.

CG a single tree

To explain the general idea of transforming the solution given by algorithm ET into a solution of the labelled augmentation problem, we first assume that CG is a single tree; later we discuss how to handle the case of CG a forest.

Definition 4.2: Let E^* initially be the set of edges added to CG by algorithm ET, and let CG^* be the graph resulting from adding E^* to CG.

We note the following

Fact 4.3: When CG is a single tree, every edge in CG is in a cycle in CG^* containing exactly one edge of E^*. That is, let C(e) be the unique cycle created by adding an edge e of E^* to CG, and let CC be the set of all such cycles, then every edge in CG is contained in at least one cycle of CC. When edge e' of CG is in C(e), we say that edge e *covers* edge e'.

Theorem 4: E^* can be transformed into an optimal solution to the labelled augmentation

problem in time $O(|X|^2)$.

Proof: If E^* is a solution to the labelled augmentation problem, then it is an optimal solution, since $|E^*| = p/2 = (r+c+b)/2$. If E^* contains no (R,R) edge or (C,C) edge then it is a solution to the labelled augmentation problem. However, if there is an (R,R) edge, e, in E^*, then since $r \leq c$, there must also be either a (C,C) edge or a (C,B) edge in E^*. Suppose there is a (C,C) edge, e' in E^*; the case of (C,B) is similar. Let S be the unique smallest subgraph of CG that connects the endpoints of e and e'. These four endpoints are leaves of S, and the topology of S is one of the three cases shown in Figure 4. In each of the three cases there is a way to add two (R,C) edges between the endpoints of e and e' so that every edge of S is in a cycle consisting of edges of S plus exactly one of the two new edges, i.e. every edge in S is covered by one of the new edges. For example, in the first case we add an edge between the upper left C and lower right R, and an edge between the upper right C and the lower left R. Now in CG^*, all edges covered by e and e' are in S, and so by fact 4.3, every edge in CG not in S is covered by some edge of $E^* - \{e, e'\}$. Hence we can delete e and e' from E^*, and add the appropriate two new (R,C) edges to E^*, so that every edge in CG is again covered by some edge in E^*. Note also that E^* remains disjoint from the edges of CG. We call such a set of edge deletions and insertions an *edge exchange*. It is clear that by successive edge exchanges we can find a set of edges, E^*, containing no (R,R) edges, such that every edge in CG is covered by an edge in E^*.

Suppose now that E^* contains no (R,R) edges; we will remove all the (C,C) edges in E^* using edge exchanges similar to those above. If there is a (C,C) edge in E^*, but no (R,R) edge, then there must be either an (R,B) or a (B,B) edge in E^* (this follows by arithmetic from the assumptions that $r \leq c$ and $r+b \geq c$). We can again do an edge exchange to replace a (C,C) and, say, a (B,B) edge with two appropriate (C,B) edges, maintaining the property that every edge of CG is covered by an edge of E^*. We repeat until E^* contain no (R,R) or (C,C) edges,

and note that E^* has throughout been disjoint from the edges of CG. Hence we have an optimal solution to the labelled augmentation problem.

To analyze the time needed for the transformation, note that at at most (r+c)/2 edge exchanges are needed, and the cost of each is bounded by the number of edges in CG. Both (r+c) and the number of edges in CG are bounded by |X|, hence $O(|X|^2)$ time suffices. Note that this could be reduced to O(|X|) time if in every edge exchange both of the possible pairs of edge additions maintain the cover of the edges of S. However, as shown in the first case of Figure 4, this is not true. □

4.1.3. Labelled augmentation when CG is a forest

In general, G may not be connected and so CG will not be a single tree; We now solve the labelled augmentation problem in the case that CG is a forest. The idea is that we will first add a set of node disjoint edges (i.e. no node is incident with more than one edge in this set) between leaves to connect the trees of CG into a single tree, T^*, and then solve the problem as before. However, we must be careful in the way that we add edges to form T^*. We call the first edge additions (forming T^*) phase one, and the next edge additions (done by algorithm ET) phase two, which is then followed by any needed edge exchanges. We let E^* now start with only those edges added in phase two. We must be careful about which edges are added in phase one, because when CG is a forest, Fact 4.3 is no longer true, i.e. it is not true that in CG^* every edge of CG is in some cycle containing only one edge of CG^* - CG (see Figure 5). Therefore the edge exchange operation and argument become much more complicated if we ever remove edges added in phase one. To get around this problem, we want to be sure that the edges added in phase one can appear together in an optimal solution to the labelled augmentation problem. If so, then these edges become fixed after phase one, and the remaining problem is exactly the labelled augmentation problem for the single tree T^*.

Theorem 5: The trees of CG can be connected (forming T*) by node disjoint (phase one) edges such that these phase one edges can appear together in an optimal solution of the labelled augmentation problem for the forest CG.

Proof: We first arbitrarily relabel (b+c-r)/2 of the B leaves in CG to be R leaves, and relabel the remaining (b+r-c)/2 B leaves to be C leaves. The effect is that CG now contains an equal number of R and C leaves and no B leaves. We now claim that it is possible to connect CG into a single tree, T*, using a set of node disjoint (R,C) edges. In fact, this can be done myopicly: as long as CG is not connected there must exist a two components in CG such that one contains an R leaf and one contains a C leaf. To see this suppose that CG is not connected, and let CC be a component containing an R leaf. Now the number of R and C leaves are equal before any phase one edge additions and this property is maintained with each edge addition, so there must also be remaining C leaves. If any of these C leaves are in a different component than CC, then an edge can be added between an R leaf in CC and a C leaf out of CC. Every component must have at least one leaf, hence if all C leaves are in CC then an edge can be added between a C leaf in CC and an R leaf out of CC.

After T* is formed, the number of R and C leaves are still equal, hence the number of (R,R) edges added by algorithm ET (in solving the unlabeled augmentation problem on T*) must equal the number of (C,C) edges added. Edge exchanges strictly between these (phase two) edges can then be done to form a legal and optimal solution to the labelled augmentation problem for CG. □

Theorem 5 gives one way to connect CG so that the phase one edges can appear together in an optimal labelled augmentation of G. In the next section it will be useful to have other ways to do this. Lemma 4.1 below gives a sufficient condition to guarantee that a set of phase one edges has this property.

Definition 4.3: Let r^*, c^* and b^* denote the number of R, C and B leaves in T^*.

Definition 4.4: Let PE denote the set of (R,C), (R,B) (C,B) and (B,B) edges running between leaves of CG.

Lemma 4.1: If the trees of CG can be connected (forming T^*) by node disjoint (phase one) edges from PE such that $r^* \leq c^*$ and $r^* + b^* \geq c^*$, or such that $c^* \leq r^*$ and $c^* + b^* \geq r^*$, then these phase one edges can appear together in an optimal solution of the labelled augmentation problem for the forest CG.

Proof: If r^*, c^* and b^* are as given in the hypothesis above, then by Theorem 4, the labelled augmentation problem on T^* can be solved optimally using $(r^*+c^*+b^*)/2$ edges from PE. Since the phase one edges are node disjoint and selected from PE, they can be added to the $(r^*+c^*+b^*)/2$ edges of the optimal solution of labelled augmentation problem on T^*, to form a labelled augmentation of CG using exactly $(r+c+b)/2$ edges from PE. We know that $(r+c+b)/2$ is a lower bound on the number of edges needed to augment CG, hence this augmentation is optimal. □

4.1.4. Labelled augmentation when $r+b < c$

We briefly discuss the labelled augmentation problem in the case when $r+b < c$. We assume that $r+b \geq 1$ since the complementary suppression problem is simple to solve directly when CG only has C leaves. When $r+b < c$, at least c additional edges must be added. If CG is a single tree, we can achieve this bound by taking the solution given by ET (which contains p/2 edges) and make edge exchanges until the only edges in E^* are (R,C) (C,B) and (C,C) edges. This can always be done since $r+b < c$. At this point we can arbitrarily pick an R or B node, v, in CG; we delete all (C,C) edges from E^* and attach each exposed C leaf to v (there are other ways to do this). In the case that CG is a forest, a similar argument to those above shows a way to

connect the forest with phase one edges which have the property that they can be together in an optimal solution. As before, the problem then reduces to the case of a single tree. Unlike the previous case, however, the phase one edges need not all be node disjoint, since even in the case of a single tree, as many as c-(r+b) edges may touch nodes that are touched by other edges of the augmentation.

Figure 6 shows a complete example of complementary suppression when D is strictly positive. Since any cycles in F* are traversable when D is strictly positive, we only indicate which cells have been suppressed, and can ignore all numerical values.

4.1.5. Labelled augmentation and necessary conditions for protection

Cox in [Cox80] discusses the utility of using simple necessary conditions for protection in heuristic methods for complementary suppression. In particular, he discusses the necessary condition that every row or column containing a sensitive cell must contain at least two suppressed cells. This is just the condition that G contain no leaves. A stronger necessary condition that subsumes this is that G contains no bridges (ignoring directions of edges), i.e. that every edge in G (or F*) is in a simple cycle (ignoring the directions of the edges) in G (or F*). When D is strictly positive, every cycle in F* is traversable, and so the necessary condition is also sufficient. In the general case this is not true, but the above labelled augmentation methodology can still be used to add the fewest number of complementary suppressions to satisfy the necessary condition that every edge in G be in a simple cycle (ignoring directions). Hence this stronger necessary condition can be efficiently and optimally imposed, and therefore may be useful in practice.

4.2. Special case: X strictly positive

Above we solved the complimentary suppression problem when D is assumed to be strictly positive. A less restrictive and more useful assumption is that the original sensitive cells are strictly positive (no zero valued data is sensitive), but other cells in D may have zero value. We do not know how to solve this problem efficiently, but under the assumption that the zero valued cells are not extremely dense, the above methodology may often be used to efficiently produce optimal solutions, or near optimal solutions.

For this discussion we assume again that, in CG, $r \leq c$ and $r+b \geq c$. Recall that suppressing a non-zero cell corresponds to adding an undirected edge to CG, and that suppressing a zero valued cell corresponds to adding a directed edge to CG. The key observation for the case of X strictly positive is that if undirected edges can be found to connect CG (forming T^*) then the resulting labelled augmentation problem on T^* can be solved optimally, i.e. no problem is caused if E^* contains directed edges. To see this, recall that no edges of T^* are deleted by edge exchanges, and that each edge in T^* is in some cycle with exactly one edge of E^*. If T^* contains only undirected edges then every such cycle is traversable since it contains at most one directed edge. Hence to optimally solve complementary suppression when X is positive, we want to connect CG (forming T^*) using edges which can be added to the optimal labelled augmentation of T^*, to form an optimal labelled augmentation of CG. Lemma 4.1 gives a sufficient condition on the selection of phase one edges which does this; Theorem 5 gives one specific way to satisfy the sufficient condition, but there are other ways as well. Since there is a fair amount of flexibility in the selection of phase one edges, if zero valued cells are not dense, then it may often be possible in practice to find a set of phase one edges that satisfy the sufficient condition, and in which case the complementary suppression problem for X positive can be solved efficiently and optimally.

If there doesn't exist a set of phase one edges satisfying the sufficient condition, or such a set is

hard to find, CG can be connected by undirected edges between leaves and interior nodes, or between two interior nodes. In the first case the resulting solution will be off the optimal by at most one plus the integer part of half the number of such edges used; in the second case the resulting solution will be off by at most the number of such edges used. With such phase one edges, the suppression problem for X positive may often be solved near optimally in practice.

In [G] we present an alternative solution to the labelled augmentation problem which has the property that when CG is a forest, the set of phase one edges are less constrained than in the solution here. In particular, we do not need to add phase one edges until CG is connected, rather we only need to add edges until a certain property holds, and this property holds if CG is connected. The most general form of the property is difficult to present here; we will only state a fairly weak sufficient condition for the property.

For the case of X positive, if CG has the property that in every component of CG there are both R and C leaves, then the labelled augmentation problem on CG can be solved optimally and efficiently. If this property does not hold in CG, then we can either add undirected phase one edges from PE, or relabel B leaves to be R or C leaves, so that in the resulting graph, G*, every component does contain both an R and a C leaf. The labelled augmentation problem on CG* can then be optimally and efficiently solved by the techniques in [G]. In applying these two operations (relabelling and adding phase one edges) the goal is to create CG* such that the optimal labelled augmentation of CG* can be added to the phase one edges to obtain an optimal or near optimal labelled augmentation of CG. If we define r*, c* and b* as the number of R,C and B leaves in CG*, Lemma 4.1 again gives a sufficient condition on r*, c* and b* so that an optimal labelled augmentation of CG is obtained in this two step manner.

4.3. Caveat

We defined a legal solution as any *non-negative* integer solution of the system of equalities given in Definition 1.1, and said that a suppressed cell (i,j) is protected if x(i,j) can take on two different non-negative integer values. Hence (i,j) is protected even if zero and one are the only two possible non-negative integer values that x(i,j) can take on. Now in the section above, we assumed that there were special constraints on D. If the adversary knows these constraints, then the definition of protection may not be relevant since the adversary may be able to use the additional information. For example in the case above, if the adversary knows that $D(i,j) > 0$, then the adversary can deduce that $D(i,j) = 1$.

The point to keep in mind is that the census bureau may use the above suppression methodology in the special cases, provided the adversary cannot know when each such case arises. The adversary may compute the possible values for cell (i,j) with the assumption that it, or other cells, are non-zero, but with the given problem set-up, the adversary doesn't know if this is a correct assumption or not. The definition of protection is relevant as long as the adversary has to work in the space of non-negative integers, even though the census bureau may sometimes make more restrictive assumptions.

If there are publicly known bounds on individual cell values, then the model and the results in this paper can be appropriately modified. For example, if it is publicly known that D is strictly positive, then the definition of a legal solution must enforce the constraint that $x(i,j) > 0$; problems a) and a') are solved virtually as before, and the solvable special case of complimentary suppression becomes the case that $D(i,j) > 1$ for all cells.

5. Interval Estimation: Tightest general upper and lower bounds

Recall that T is the value of the maximum flow in G. As mentioned earlier, the tightest lower and upper bounds on the value of a suppressed cell (i,j) are given by the minimum C(i,j) and maximum L(i,j) that permit a flow of value T in G. In this section we show how to compute these minimum and maximum values.

5.1. Computing the minimum C(i,j)

If we let C(i,j) be a parameter, we can express the value of the maximum s-t flow in G as a function of C(i,j), assuming that all other capacities are fixed as given in definition 2.2. Letting $T_{i,j}(0)$ and $T_{i,j}(\infty)$ be respectively the maximum flow values when C(i,j) is set to zero and to infinity, it is known [FF] that the maximum flow, as a function of C(i,j), exactly equals $\text{Min}[T_{i,j}(0) + C(i,j), T_{i,j}(\infty)]$. However, $T_{i,j}(\infty)$ clearly equals T, and so

Theorem 6: The tightest lower bound on the value of cell (i,j) is $T - T_{i,j}(0)$, i.e. $T\text{-}T_{i,j}(0)$ units are needed on edge <i,j> and there exists a flow of value T in which edge <i,j> has exactly $T\text{-}T_{i,j}(0)$ units.

This result is simple, yet it leads (in worst case analysis) to a computational improvement over earlier suggested methods based on *minimum cost* network flow or linear programming. The point is that while there does exist a flow of value T in which <i,j> has flow $T - T_{i,j}(0)$, we don't need to find such a flow; we only need to compute $T_{i,j}(0)$. Previous approaches found the minimum permitted F(i,j) by using minimum cost flow or linear programming to obtain a maximum flow in which the minimum F(i,j) is achieved. These methods are more complex than simple network flow used to compute $T_{i,j}(0)$. Notice that we can't generally construct a flow of value T by taking the flow used to compute $T_{i,j}(0)$, and then adding $T - T_{i,j}(0)$ units along the path s-i-j-t, since adding these units might result in a violation of the capacity constraint of

either of the edges <s,i> or <j,t>. However, once $T_{i,j}(0)$ is known, we can find a maximum flow in G in which F(i,j) is at its minimum value, using only one additional maximum flow computation: simply delete <i,j> from G, reduce both C(s,i) and C(j,t) by T - $T_{i,j}(0)$, and find a maximum flow in the resulting graph; a flow of T in which F(i,j) is at its minimum is then obtained by adding T - $T_{i,j}(0)$ units along the s-i-j-t path.

We note that the above lower bound calculations are needed for at most n+m-1 cells. From the theory of linear programming it is known that in any "basic" maximum flow F in G, at most n+m-1 of the center edges have non-zero flow. A basic maximum flow can be found by one of many methods that are fast in practice [MUR], and such a flow identifies at least nm-n-m+1 cells with tight lower bounds of zero.

5.2. Computing the maximum L(i,j)

To find the tightest upper bound on the value of cell (i,j) we consider L(i,j) as a parameter and compute the maximum value L(i,j) can take on such that a flow in G of value T is still possible. Equivalently we consider the graph G′ resulting from deleting edge <i,j> from G and replacing the capacities of edges <s,i> and <j,t> with the parameterized capacities of C(s,i) - L(i,j) and C(j,t) - L(i,j), where C(s,i) and C(j,t) are fixed, and L(i,j) is variable.

Definition 5.1: Define the function F[L(i,j)], for values of L(i,j) between zero and Min[R*(i), C*(j)], to be the value of the maximum flow in G′ as a function of L(i,j).

Then, the following is immediate.

Lemma 5.1: For any value of L(i,j), there exists a flow of value T in G in which F(i,j) = L(i,j), if and only if there exists a flow of value T - L(i,j) in G′, i.e. F[L(i,j)] = T - L(i,j).

It will be convenient to work with G′ instead of G. We want to find the maximum value of

L(i,j) such that F[L(i,j)] = T - L(i,j); by lemma 5.1 that is the maximum value of L(i,j) such that there is a flow in G of T and F(i,j) = L(i,j), and hence it is the tightest upper bound on the value of cell (i,j). We make the following claims.

Lemma 5.2: F[L(i,j)] is a piecewise linear function with slopes 0, -1 or -2.

Proof: As a function of L(i,j), the capacity of every s-t cut in G′ is a straight line with slope of either 0, -1 or -2 (depending on the number of edges with parameterized capacities that cross the cut), and for any value of L(i,j), F[L(i,j)] equals the minimum value of all the s-t cuts in G′. □

Lemma 5.3: $F[L(i,j)] \leq T - L(i,j)$ for all values of L(i,j) and equality holds for $L(i,j) = T - T_{i,j}(0)$.

Proof: First, $F[L(i,j)] \leq T - L(i,j)$, since otherwise there would be a value of L(i,j) permitting a flow in G of value greater than T. Second, from Theorem 6 we know that there exists a flow F in G of value T in which $F(i,j) = T - T_{i,j}(0)$, hence there is a flow in G′ of value $T - (T - T_{i,j}(0)) = T_{i,j}(0)$ when L(i,j) is set to $T - T_{i,j}(0)$. So $F[L(i,j)] = T - L(i,j)$ when $L(i,j) = T - T_{i,j}(0)$. □

Now by definition, $F(0) = T_{i,j}(0)$, and so lemmas 5.2 and 5.3 imply that F[L(i,j)] has the shape given in Figure 7.

Definition 5.2: Let $L^*(i,j)$ be the value of L(i,j) at the intersection of the line T - L(i,j) and the line supporting the segment of F[L(i,j)] with slope -2 (see Figure 7).

Given the shape of F[L(i,j)], the tightest upper bound on the value of cell (i,j) is clearly $\text{Min}[L^*(i,j), R^*(i), C^*(j)]$. We can compute $L^*(i,j)$ if we know the equation of the line supporting the segment of F[L(i,j)] with slope -2. That equation is obtained from the observation that the cut associated with the line is simply the cut of least capacity that contains both edges <s,i>

and <j,t>. We can find this cut by deleting <s,i> and <j,t> from G and computing the minimum s-t cut in the resulting graph; let $T_{i,j}(0,0)$ denote the value of that cut. Then, as a function of L(i,j), the least capacity cut in G′ containing <s,i> and <j,t> has capacity $T_{i,j}(0,0) + R^*(i) + C^*(j) - 2L(i,j)$, hence $L^*(i,j) = T_{i,j}(0,0) + R^*(i) + C^*(j) - T$.

Summarizing the above, we get

Theorem 7: The tightest upper bound on the value of suppressed cell (i,j) is $\text{Min}[R^*(i), C^*(j), T_{i,j}(0,0) + R^*(i) + C^*(j) - T]$.

The computational import of this theorem is that one network flow computation suffices to compute $T_{i,j}(0,0)$ and hence to compute the tightest upper bound on the value of (i,j). As in the case of the lower bound, we do not need to actually find a flow of value T in which the maximum flow in edge <i,j> is achieved, although such a flow could be found with one additional maximum flow computation.

Figure 8 illustrates the above estimation methodology.

6. Tightest upper and lower bounds in a totally suppressed table

In [COX80] the case of a totally suppressed table (all cell values are suppressed, but row and column totals are disclosed) is discussed, and it is suggested that algorithms for the transportation problem can be used for this case. We show here that, in this case, trivial methods suffice to compute the tightest upper and lower bounds on the cell values.

Theorem 8: In a totally suppressed table the tightest upper bound on the value of cell (i,j) is Min[R(i), C(j)].

This can be proved by specializing Theorem 7, but we prove it here using a lemma that will be useful in the discussion of the lower bounds.

Lemma 6.1: Let R(1),...,R(m) and C(1),...,C(n) be any non-negative integers such that $\sum_i$ R(i) = $\sum_j$ C(j). Then there exists a legal solution to the table D where all cells are suppressed and R(i) is the i'th row total and C(j) is the j'th column total in D.

Proof: In the network flow interpretation of table D above, graph H is a complete bipartite graph, and so in G any s-t cut must contain either all edges incident with s, or all edges incident with t. Hence the minimum cut and maximum flow have value Σ R(i), and there is a legal solution to D. □

Proof of Theorem 8: Clearly Min[R(i), C(j)] is an upper bound on the value of cell (i,j). Assume Min[R(i), C(j)] = R(i). To show that the bound can be met we must exhibit a solution to D in which (i,j) is given value R(i). We simply give cell (i,j) the value R(i), and give all other cells in row i the value 0; we then remove row i from D and set C(j) to C(j) - R(i). The row totals still equal the column totals in the reduced table, and so by lemma 6.1, the reduced table has a solution, and hence Min[R(i), C(j)] is an attainable value for (i,j). □

Now we examine the lower bound.

Theorem 9: In a totally suppressed table the tightest lower bound on the value of cell (i,j) is Max[0, R(i) + C(j) - T].

Proof: We first show that the above is a lower bound. The cell values in row i must add up to R(i), but without cell (i,j), the total that all other cells in row i can contribute is T - C(j). Hence cell (i,j) must have value at least R(i) + C(j) - T. The same lower bound is obtained by doing the analysis along column j. To show that this bound is tight, give cell (i,j) the value Max[0, R(i) + C(j) - T], and assign values to the other cells in row i so that the total assigned in row i is exactly R(i) (this is always possible). After deleting row i and reducing each C(k) by the amount assigned to cell (i,k), the row and column totals are still equal, and so, by lemma 6.1, the reduced

table has a solution, and the theorem is proved. □

Corollary 6.1: In a totally suppressed table of size n by m, cell (i,j) can have a non-zero lower bound only if either R(i) > T/2 or C(j) > T/2; there are at most n+m-1 such cells.

Proof: Max[0, R(i) + C(j) - T] = 0 unless R(i) + C(j) > T, which can happen only if either R(i) > T/2 or C(j) > T/2. Since $\sum_i R(i) = \sum_j C(j) = T$, at most one R(i) and one C(j) can be greater than T/2. Hence any non-zero lower bounds are for cells contained in one specific row or one specific column, and there are only n + m - 1 of such cells. □

Hence we need to compute the lower bound only for those (at most) n + m - 1 cells; all other cells have lower bound of zero. Therefore O(n+m) arithmetic operations and comparisons suffice in this case, although the output is of size O(nm). Note that we already knew from the discussion in section 5.1 that at most n + m - 1 lower bounds would be non-zero, but in the general case it is more difficult to identify those cells.

Corollary 6.2: Any n by m totally suppressed table is protected, for n,m > 1.

Proof: Cell (i,j) is protected if Min[R(i), C(j)] > Max[0, R(i) + C(j) - T]. We assumed that every row and column sum was non-zero, so (i,j) is protected if Min[R(i), C(j)] > R(i) + C(j) - T. Assume that R(i) ≤ C(j). If R(i) = R(i) + C(j) - T then C(j) = T, but when n > 1 this is a contradiction to the assumption that all column sums are non-zero. □

7. Open Questions and Future Research

There are many interesting open questions remaining. We believe that many of these questions are solvable and hope that the ideas in this paper will be useful in their solution.

1. The computation of lower and upper bounds for the general two dimensional table involves many successive maximum flow computations, each on a graph which differs from the other

graphs by a small modification. Can this be exploited to compute all of the maximum flows faster than computing them all from scratch?

2. Are there additional interesting special cases of the complimentary suppression problem that can be efficiently and optimaly solved? Is the general problem efficiently solvable? Is it NP-hard. What about bounded approximations? It is not hard to show that the general problem can be approximated to within a factor of two of the optimal.

3. In [G] we examine the augmentation problem on mixed graphs; this is one step towards the solution of the general complimentary suppression problem, however the problems solvable by the methods in [G] do not include the general problem. Can these methods by pushed to solve augmentation problems that are closer to the general complimentary suppression problem?

4. Generalize the approach in this paper to tables of three or more dimensions. Even the totally suppressed case is open for three dimensions. For example, the three dimensional analogues of theorems 8 and 9 don't even hold (see Figure 9).

5. In this paper a cell was defined to be protected if the tightest upper and lower bounds differ by at least one. A more useful condition is that they should differ by some $\delta > 1$. Can we efficiently (linear time) determine whether D is protected for a given $\delta > 1$ specified ahead of time? What about for δ given as input? What about the complementary suppression problem for in these cases? These problems are the most tempting and promising of this list.

6. It may be realistic to assume bounds on the sums of a set of cell values, or correlations between the values of certain cells. How can these be handled in the protection and suppression and estimation problems?

7. In the special case of X strictly positive, can we efficiently form CG* using edges which satisfy the conditions of Lemma 4.1? If undirected edges are plentiful then in practice this will

often be easily done, but what is the complexity of the exact optimization problem? What is the complexity of the problem of adding node disjoint undirected edges (from a restricted set of edges) between leaves to connect a set of trees?

9. The approach to the complementary suppression problem in this paper and also in [G] is to modify and build on the two algorithms given in ET for unlabeled (undirected and directed) graph augmentation. There are surely other solutions to these unlabeled augmentation problems, and each may have a different consequence for the complementary suppression problem, leading, for example, to different solvable special cases or different heuristics. This is an area for future research.

8. Acknowledgement

I would like to thank everyone in the Yale theory group and Dick Karp for listening to parts of this work as it evolved. Special thanks to Gregory Sullivan for reading the penultimate draft.

9. References

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

[COX75] Cox, Lawrence. "Disclosure Analysis and Cell Suppression", Proceedings of the American Statistical Association, Social Statistics Section, 380-382, 1975.

[COX77] Cox, Lawrence. "Suppression Methodology in Statistical Disclosure", Proceedings of the American Statistical Association, Social Statistics Section, 750-755, 1977.

[COX78] Cox, Lawrence. "Automated Statistical Disclosure Control", Proceedings of the American Statistical Association, Survey Research Methods Section 177-182, 1978.

[COX80] Cox, Lawrence. "Suppression Methodology and Statistical Disclosure Control",

Journal of the American Statistical Association, Theory and Methods Section, June 1980, Vol. 75, Number 370.

[DEN] Denning, Dorothy. Cryptography and Data Security. Addison- Wesley 1982.

[ET] Eswaran, K. and Tarjan, R. E. "Augmentation Problems", SIAM J. Computing, Vol. 5, No. 4, December 1976.

[EVE] Even, Shimon. Graph Algorithms, Computer Science Press, 1979.

[FF] Ford, L. and Fulkerson, D. Flows in Networks, Princeton University Press, 1962.

[G] Gusfield, Dan. "Optimal Mixed Graph Augmentation", Yale University department of Computer Science Technical Report # Yale/DCS/TR-327, August 1984.

[MUR] Murty, Katta. Linear and Combinatorial Programming, Wiley, 1976.

T = 31		4	5	4	6	7	C*(j)
	73	18	6	17	15	17	C(j)
7	21	[5]	[2]	6	[0]	8	
4	10	[4]	[0]	0	4	2	
7	15	3	[3]	[4]	5	[0]	
7	13	4	0	2	[0]	[7]	
6	14	2	1	5	[6]	[0]	
R*(i)	R(i)		Table D				

Figure 1a:

The squared numbers indicate the sensitive cells.

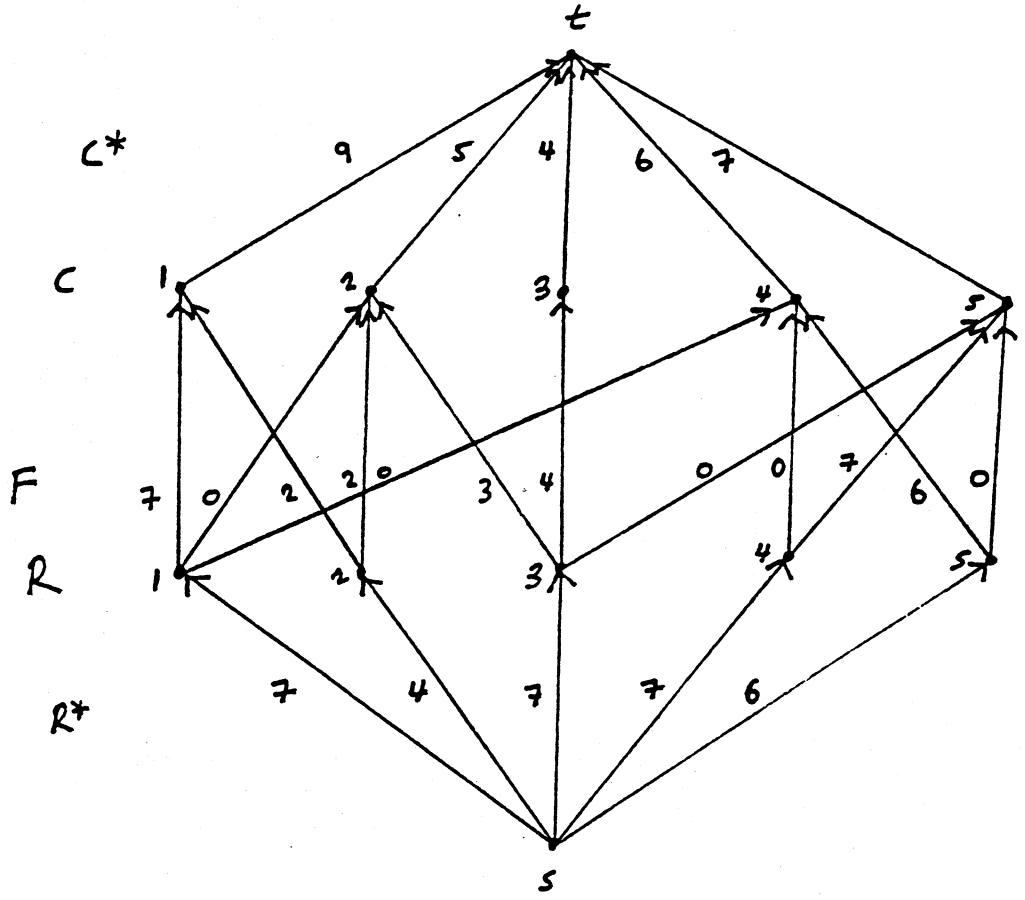

Figure 1b:

Graph G derived from D. A maximum flow F is indicated on the center edges.

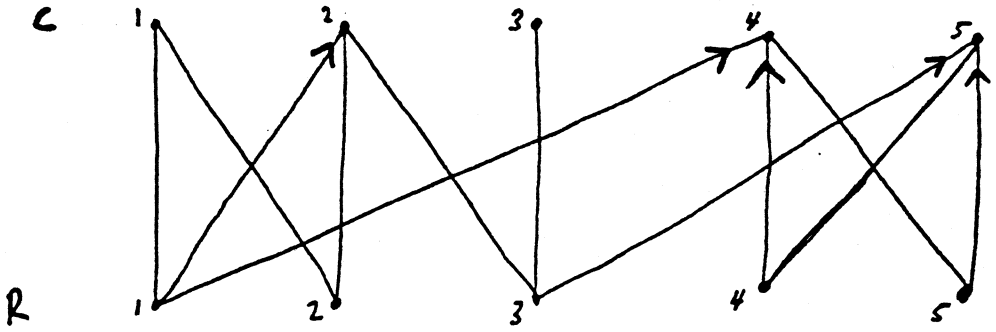

Figure 2:
Graph F* derived from G and the flow F given in Figure 1.

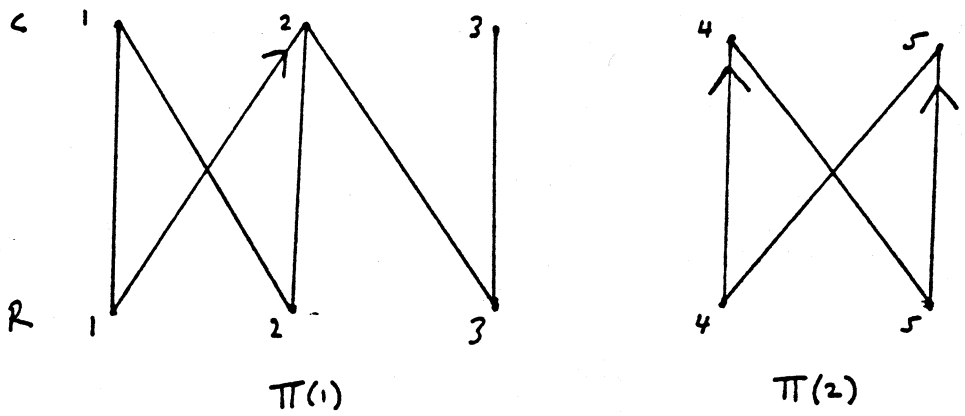

Figure 3a

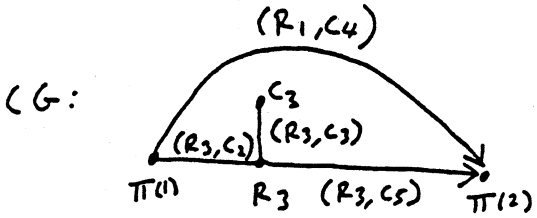

Figure 3b

Figure 3:

Construction of graph CG from F*
Figure 3a shows the two strong components $\pi(1)$ and $\pi(2)$ of F*. In $\pi(1)$ the bridges are (R_3, C_2) and (R_3, C_3). There are no bridges in $\pi(2)$. The edges of set A are (R_1, C_4) and (R_3, C_5).
Figure 3b shows graph CG.

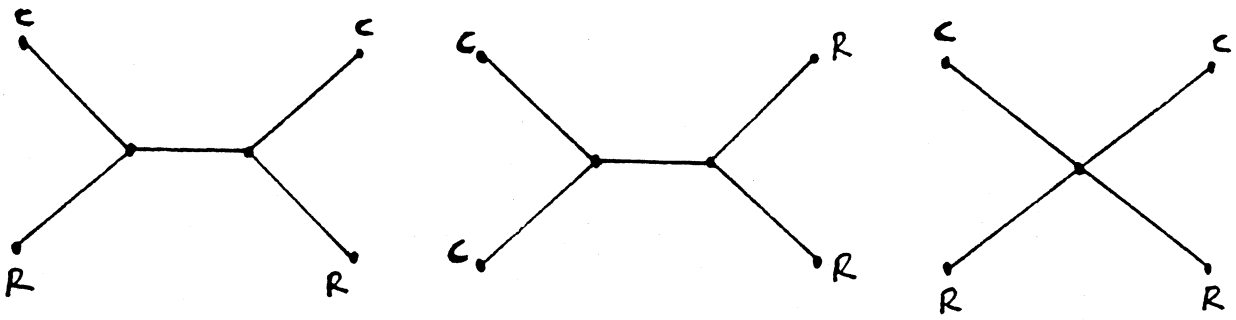

Figure 4:

Cases for edge exchanges. Each line represents a path between the endpoints.

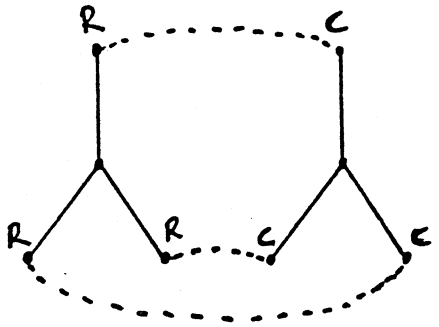

Figure 5:

CG is shown in solid and the optimal augmentation is dashed. No edge in CG is covered by just a single edge of the optimal solution.

D

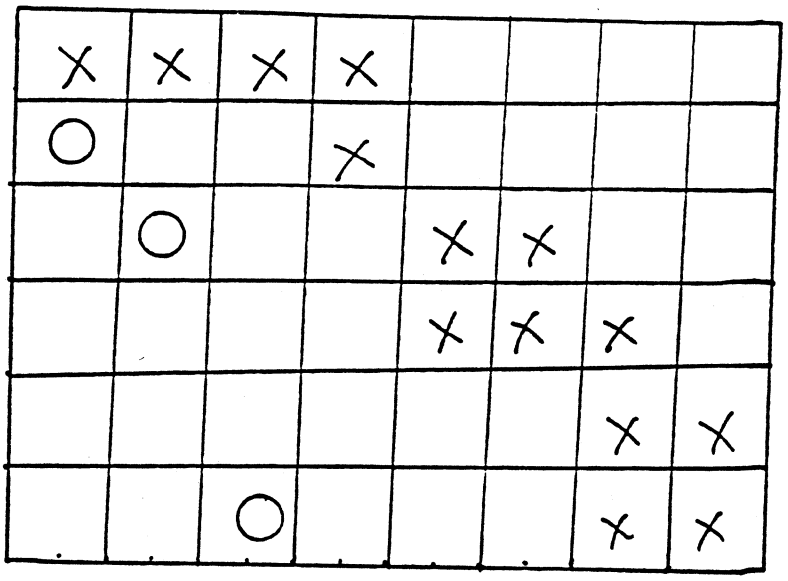

Figure 6e:
Table D with complementary suppressions added.

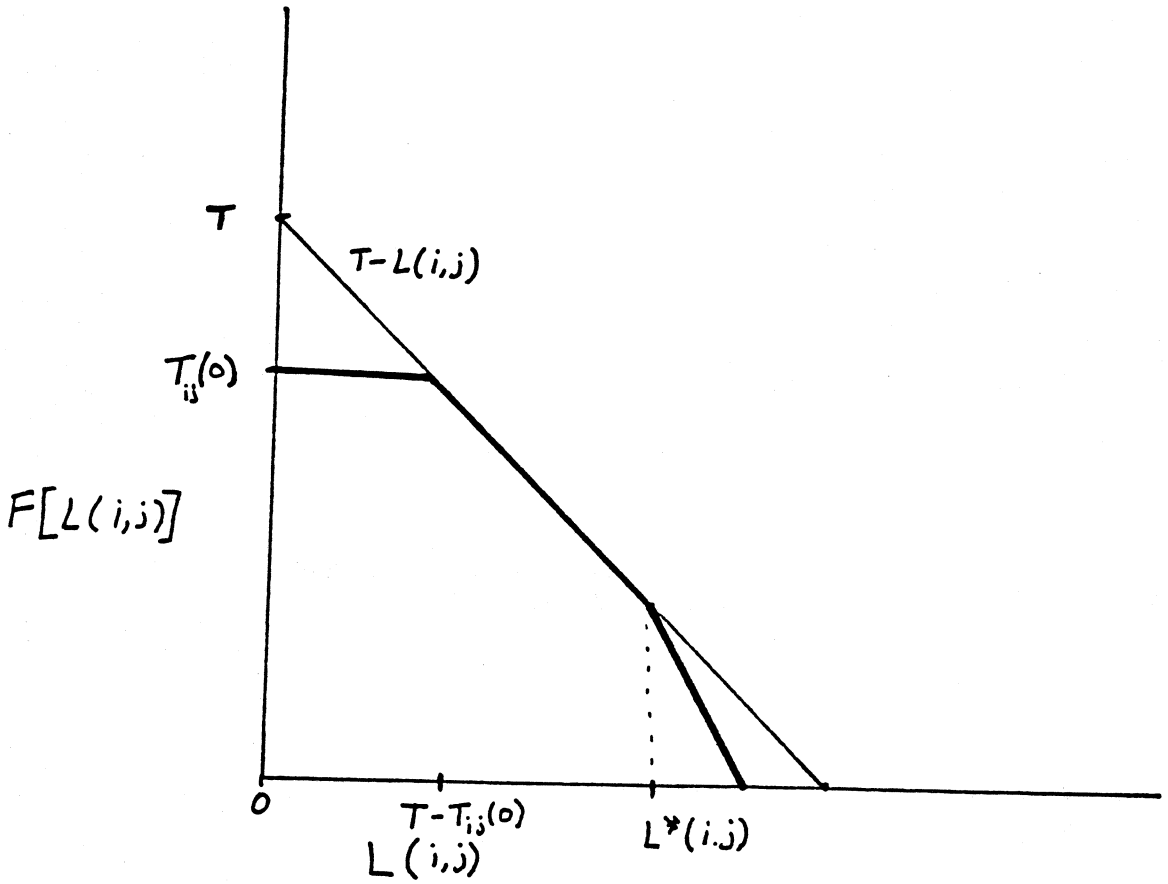

Figure 7:
The general shape of F[L(i,j)].

D

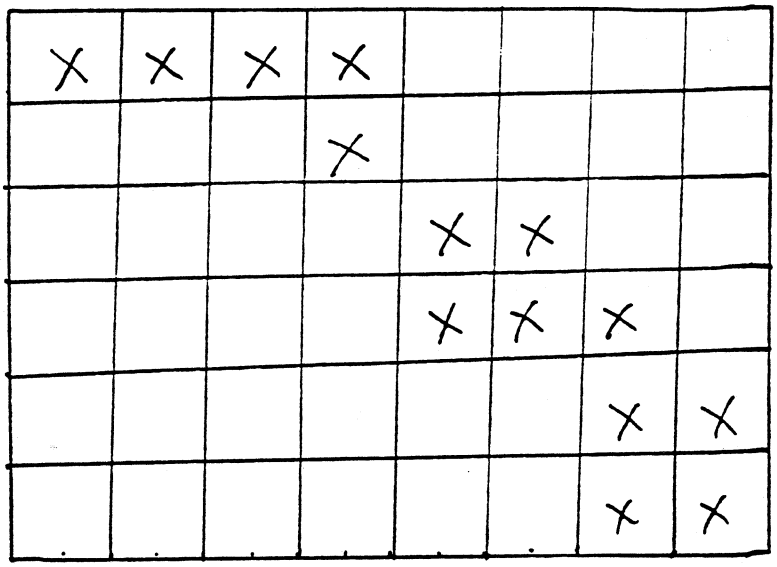

Figure 6a:

Example of complimentary suppression. D is assumed strictly positive; the cells with x's are suppressed.

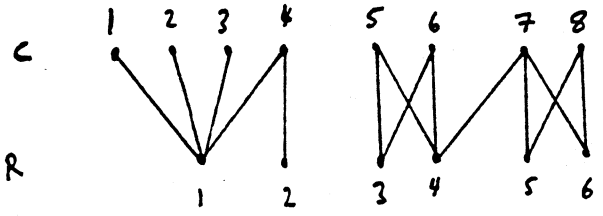

Figure 6b:

Graph F* derived from D.

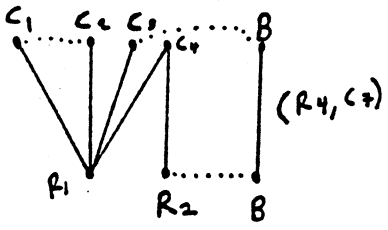

Fig. 6d

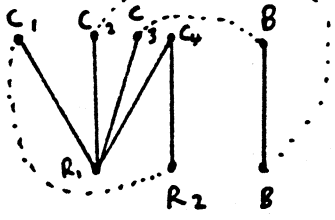

Figure 6c:

Graph CG derived from F* with phase one and two edges (dashed) added. Edge (C_3, B) is the phase one edge.

Figure 6d:

Solution to the labelled augmentation problem after edge exchange.

T = 58

		12	24	22	c*(j)
	75	20	30	25	c(j)
19	25	X	6	X	
22	30	8	X	X	
17	20	X	X	3	

D:

R*(i) R(i)

Figure 8a:

Example for interval estimation (taken from p. 363 of [Den]). The cell with x's are suppressed. T = 58.

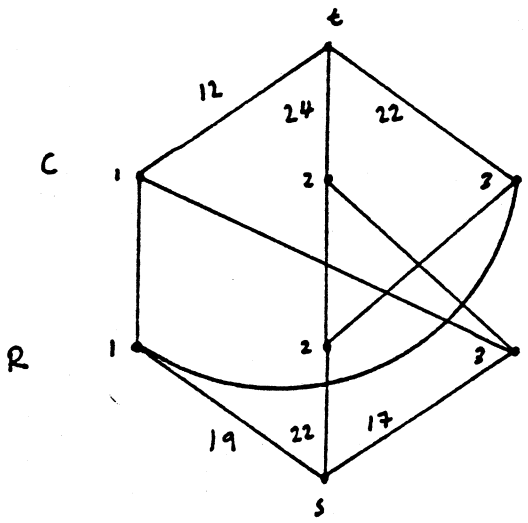

Figure 8b:

Graph associated with table D. We compute the upper and lower bounds on cell (2,2). Removing edge (R_2, C_2), the maximum flow is 51, so the best lower bound on cell (2,2) is 58-51=7. Removing edges (s, R_2), (R_2, C_2), and (C_2, t) the maximum flow is 31, so the best upper bound on cell (2,2) is 31+22+24-58=19.

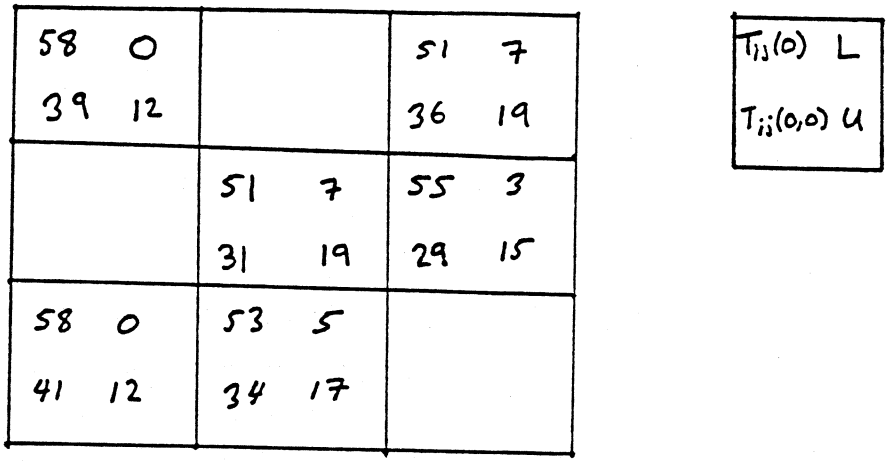

Figure 8c:

Mximum flow values for the graph in figure 8b, and upper and lower bounds on the suppressed cell values of table D. In each square the upper left entry is $T_{i,j}(0)$, the upper right entry is $T-T_{i,j}(0)$, the lower left entry is $T_{i,j}(0,0)$, and the lower right entry is $\text{Min}[R^*(i), C^*(j), T_{ij}(0,0) + R^*(i) + C^*(j) - T]$. The reader can check that these bounds agree with those on p. 364 of [Den].

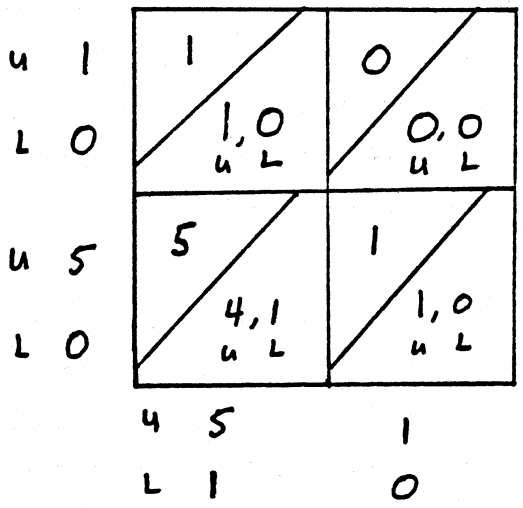

Figure 9:

Three dimensional table of size 2x2x2. The two numbers in each cell below the triangle are the upper and lower table entries respectively, i.e. cell (1,1,1) has entry 1, cell (2,1,1) has value 4, cell (2,1,2) has value 1 etc. The numbers in the triangles are the sums of the cells looking downward. The numbers on the left side are upper and lower sums of the cells looking horizontally along a line, and the numbers on the bottom are upper and lower sums of the cells looking vertically along a line. The minimum of the sums constraining cell (2,1,1) is 5, and so the " analogue " of Theorem 8 suggests that in the totally suppressed table with these sums (2,1,1) could have value 5. But this is not possible since then cell (2,2,1) would be forced to be zero and then cell (1,2,1) would be forced to be 1, which contradicts the fact that cell (1,2,1) is forced to be zero by the zero in the triangle.

Addendum 2/17/85

We note two responses to the open questions listed in section 7.

1. The method given in this report for computing the tightest upper bound on the value of a suppressed cell takes $O(n^3)$ time per cell. Hence, in an n by n table, if $\Theta(n^2)$ upper bounds were to be computed, the best time guaranty based on the method in this report is $O(n^5)$. We show here that this time bound can be reduced to $O(n^4)$, and that O(nlogn) maximum flow computations suffice to find all the tightest upper bound values in any table. A consequence of this approach is that in any table (even those with $\Theta(n^2)$ missing values), there are never more than 2n-1 distinct upper bound values. These results are obtained by reducing the problem of computing upper bounds to a problem discussed in a paper by Schnorr [SC].

We first give a different method for computing a single upper bound. Let x be a legal assignment of values to suppressed cells in D (hence x defines a maximum flow F in the graph G given in definition 2.2), and let G(F) be the *augmentation* graph [FF] defined by F and G, except that the forward capacities of G(F) are set to be some large finite value M instead of infinity. Let (i,j) be a suppressed cell in D, where i is an R node in G, and j is a C node in G. Let x(i,j) be the value of cell (i,j) in the assignment x, and let FG(i,j) be the value of the maximum total flow from i to j in G(F), and let FG(j,i) be the value of the maximum total flow from j to i in G(F). Then, FG(j,i) is the tightest upper bound on the value of cell (i,j) in D, and Max[0, x(i,j) - FG(i,j) + M] is the tightest lower bound on the value of cell (i,j) in D. It is not difficult to prove these assertions, and we omit the proofs here.

The above new relations require one maximum flow computation per bound, and hence do not immediately improve on the method presented in the report. To prepare for the improvement, note two facts about this method. First, unlike the method in the report, the graph G(F) is fixed, but the choice of source and sink nodes varies over all pairs (i,j) such that (i,j) is a suppressed cell in D. Second, for every suppressed cell (i,j) in D, note that $FG(i,j) \geq M > FG(j,i)$ = tightest upper bound on the value of cell (i,j). Hence, in G(F), Min[FG(i,j), FG(j,i)] = tightest upper bound on suppressed cell (i,j), for each suppressed cell (i,j). We would like an efficient method to compute FG(i,j) for any pair of nodes i,j in G(F). Such a method is known for undirected graphs [GH]; in that method, all $\Omega(n^2)$ flow values can be computed with only n-1 flows in an n node undirected graph. However, for directed graphs, no such equivalent method is known. What is known, however, is how to compute the $\Omega(n^2)$ values of Min[FG(i,j), FG(j,i)] for all pairs of nodes in a directed graph using at most O(nlogn) maximum flows. Further, the total time needed for all the flows together can be bounded by $O(n^4)$ [SC]. Using this method on G(F),

all tightest upper bounds on the values of suppressed cells can be computed in $O(n^4)$ time. Note that for the lower bounds, this bound was already known to be attainable, since there can only be 2n-1 non-zero lower bounds, and the cells where the non-zero bounds can appear can be located quickly. Then a single flow per cell finds the value of the lower bounds. Hence $O(n^4)$ time suffices to find all tightest bounds in an n by n table. There is a easy reduction from the maximum flow problem to the problem of computing a single bound, hence $O(n^3)$ seems the best realistic target for the time needed to compute all the bounds.

We have shown that for any suppressed cell (i,j) the tightest upper bound on the value of cell (i,j) is equal to $B(i,j) = \text{Min}[FG(i,j), FG(j,i)]$ in G(F). It is known [SC] that there can never be more than 2n-1 distinct values for $B(i,j)$ in any directed graph with 2n nodes. Since the B values define the tightest upper bounds in D, there can never be more than 2n-1 distinct tightest upper bounds in D. This is easily verified on the table where all cells are suppressed, but holds in general. We give a short proof of this below.

Lemma [SC]: In any directed graph G, let I be a subset of vertices arbitrarily labelled 1,2,...,z. For any pair of nodes $i < j$ in I, if $B(i,j) = FG(i,j)$, then $B(i,j) \geq \text{Min}[B(k,k+1), k = i,...,j-1]$.

Proof: Let (S,S′) be the minimum cut separating nodes i and j. Since $i \in S$ and $j \in S'$, there must exist a pair of nodes w and w+1 in I such that $w \in S$ and $w+1 \in S'$, and hence $FG(w,w+1) \leq$ capacity of cut (S,S′). So $\text{Min}[B(k,k+1), k = i,...,j-1] \leq B(w,w+1) \leq FG(w,w+1) \leq FG(i,j) = B(i,j)$. □

Theorem: Let H be a directed graph with 2n nodes. Then there can be at most 2n-1 distinct values of B.

Proof: Consider a complete undirected graph G′ with 2n nodes and weight $B(i,j)$ on every edge (i,j). Let T be a maximum weight spanning tree of G′. T has exactly 2n-1 edges. Now consider any edge (i,j) of G′ that is not in T. Since T is a maximum weight spanning tree, $B(i,j)$ is less than or equal to every B value on the edges in the unique path between i and j in T. But the lemma above says that $B(i,j)$ must be greater or equal to the minimum of those values, hence it must be equal to the minimum of those B values. Therefore, the B value of every edge not in T is equal to the B value of some edge in T, and the theorem is proved. □

Combining the theorem with the fact that the tightest upper bounds of D are equal to the B values in G(F) gives the following

Theorem: In any n by n table D, there are at most 2n-1 distinct tightest upper bounds on the

values of the suppressed cells in D.

2. The two problems listed in open problem 7 are both NP-hard. The Hamilton path problem can be simply reduced to either of these problems.

References:

[GH] R.E. Gomory and T.C. Hu, "Multi-terminal network flows", SIAM J. applied math, 9 (1961)

[SC] C.P. Schnorr, "Bottlenecks and edge connectivity in unsymmetrical networks", SIAM J. Computing, Vol. 8, No. 2, May 1979.