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

Abstract

In this paper we consider an augmentation problem on mixed graphs that generalizes and unifies two augmentation problems, one for undirected graphs and one for directed graphs, considered by Eswaren and Tarjan [ET]. The mixed graph augmentation problem has applications in the design of communication networks, and forms of the mixed augmentation problem are central in statistical data security problems discussed in [G]. We solve the augmentation problem on mixed graphs in time linear in the number of edges of the graph, the same time bound obtained in [ET] for each of the two special cases.

OPTIMAL MIXED GRAPH AUGMENTATION

Dan Gusfield
YALEU/DCS/TR-327
August, 1984

Table of Contents

1. Introduction

In this paper we consider an augmentation problem on mixed graphs that generalizes and unifies two augmentation problems considered by Eswaren and Tarjan [ET]. The mixed augmentation problem has applications in the design of communication networks, and forms of the mixed augmentation problem are central in statistical data security problems discussed in [G].

Let G be a connected undirected graph and G′ a connected directed graph. The following two problems are solved with linear time algorithms in [ET]: add the fewest number of *undirected* edges to G so that the resulting graph contains no bridges, and add the fewest number of *directed* edges to G′ so that the resulting graph is strongly connected. The first problem is equivalent to the problem of adding the fewest number of undirected edges to G so that every edge is in an undirected simple cycle, and the second problem is equivalent to the problem of adding the fewest number of directed edges so that every edge is in a directed simple cycle. In a *mixed* graph (one with both undirected and directed edges) we say that a cycle is *traversable* if and only if it is possible to walk around the cycle without walking opposite to the direction of any directed edges on the cycle. Clearly an undirected or a directed cycle is traversable. It will follow from Theorem 1 below that the first problem is also equivalent to that of adding the fewest number of *directed* edges to the undirected graph G so that in the resulting *mixed* graph every edge is in a traversable simple cycle. The second problem can obviously also be similarly described, i.e. as the problem of adding the fewest directed edges to the directed graph G′ so that every edge is in a traversable simple cycle. Hence a common generalization of these two augmentation problems is that of adding the fewest number of *directed* edges to a *mixed* graph so that every edge in the resulting graph is in a traversable simple cycle. We call this the *mixed graph augmentation* problem; we will give a linear time algorithm to solve this problem.

The mixed graph augmentation problem is of interest not only because it generalizes the

strictly undirected and the strictly directed augmentation problems, but because its solution unifies the separate solutions presented for these two problems in [ET]. The mixed problem is also central in the study of data security [G] where a data protection problem reduces to a special case of the mixed augmentation problem. In fact, the work reported here results from an effort to solve the most general form of the protection problem. In section 3 we will briefly discuss this problem and show how the results of this paper efficiently solve a special case of it.

As a final application for the mixed augmentation problem consider the task of completing or upgrading a partially constructed communication system: one-way communication lines are to be laid out so that every point in the system can send data to any other point, i.e. the graph representing the system must be made strongly connected. Input for the problem consists of a connected[1] mixed graph G and a directed graph $\bar{G}$. The directed edges in G represent communication lines that already exist, the undirected edges in G represent existent conduits or right of way along which one-way lines can be (cheaply) added, and the edges in $\bar{G}$ represent one-way lines requiring new (expensive) conduits or right of way. In the most general problem there is a distinct cost to route a line in each conduit and a distinct cost to build each new conduit; the general problem is to choose the cheapest way to use the existing conduits and to add more. This problem is difficult to solve. However, if the cost of building each new conduit is roughly the same, and much greater than that of routing line in an existing conduit, then the following simpler problem is of interest: minimize the number of new conduits needed so that lines can be laid to make the graph strongly connected. We call this the *communication completion* problem and we will see that this is exactly the mixed augmentation problem.

---

[1]We assume G is connected for simplicity. We can extend the ideas here to the case of G disconnected.

1.1. Definitions and Problem Specification

Definition 1.1: Let G = (V,E) be a connected *mixed* graph with node set V and edge set E, where E can contain both directed and undirected edges. Let $\bar{G}$ = (V,$\bar{E}$) be a *directed* graph on the same nodes.

Definition 1.2: We use the notation <i,j> to indicate a directed edge from i to j, and (i,j) to denote an undirected edge between nodes i and j. When the distinction does not matter, we will use the unmodified term "edge" to refer to either a directed or an undirected edge. Note that the term "mixed graph" permits the special cases that all the edges of G are directed or are all undirected.

Definition 1.3: A cycle C in G is called *traversable* if and only if it is possible to walk around the cycle without ever walking against the direction of any directed edge on C. We call an edge e (directed or undirected) in G *traversable* if and only if e lies on some traversable simple cycle in G, and we call a mixed graph traversable if and only if all its edges are traversable.

Note that a cycle consisting of the two directed edges <i,j> and <j,i> is considered a simple cycle, as is one consisting of the undirected edge (i,j) and, say, the directed edge <j,i>, but the path i,j,i that traverses a single undirected edge (i,j) in both directions, is not a simple cycle.

Definition 1.4: The *mixed augmentation problem* is to find a minimum number of edges of $\bar{E}$-E, such that when these edges are added to G, every edge in E is traversable[2]. Let M(G) denote the size of the smallest such set of new edges, and define M(G) = ∞ if there are edges of G that are not traversable even when all edges of $\bar{G}$ are added to G. In the case that G is undirected (directed), the mixed augmentation problem is called the undirected (directed)

---

[2]If the solution is optimal then all the new edges will also be traversable.

augmentation problem.

Fact 0: A connected directed graph is traversable if and only if it is strongly connected. A connected undirected graph is traversable if and only if it has no bridges, if and only if it can be directed to become strongly connected (see [O] for a frivolous application of this). A connected mixed graph is traversable if and only if its edges can be directed to become strongly connected. This last fact will follow from Theorem 1 below.

When $\bar{G}$ is permitted to be an arbitrary directed graph on node set V, the directed and undirected augmentation problems are NP-complete [ET], [FJ]. Tarjan and Eswaren [ET] present two linear time algorithms to solve the undirected and directed problems[3] with the assumption that $\bar{G} = (V,\bar{E})$ is the complete directed graph, i.e. $\bar{E}$ contains the edges $<i,j>$ and $<j,i>$ for every pair of nodes i,j. We will also make this assumption and the solve the resulting mixed augmentation problem.

2. Solving Mixed Augmentation via Knapsack

In this section we assume, as in [ET], that $\bar{G}$ is a complete directed graph, and give a linear time algorithm for the mixed augmentation problem. The method uses a special form of the knapsack problem to reduce instances of the mixed problem to instances of the directed problem. The directed problem is then solved by the algorithm for the directed augmentation problem in [ET]. We first reduce the mixed problem on G to an orientation problem on G, and then to an orientation problem on a derived graph CG.

---

[3]In [ET] graph G is not assumed to be connected, but the graph after augmentation is required to be connected. We can similarly modify the techniques here to include this case but the exposition is simplified without it. The reader should be alerted that in [ET] the most critical implementation details (algorithm ST) of the method for directed augmentation are in error, but have a simple correction.

2.1. Reducing mixed augmentation to orientation

Definition 2.1: Given mixed graph G, let G′ denote a directed graph obtained by some orientation of the undirected edges of G.

Clearly, $M(G) \leq M(G')$, but Theorem 1 below shows that equality holds for some G′.

Theorem 1: There exists a directed graph G′ resulting from orienting the undirected edges of G, such that $M(G) = M(G')$.

This theorem is very useful and somewhat surprising since the definition of "traversable cycle" permits the use of an undirected edge in opposite directions on two different traversable cycles. The proof of the theorem will use the concept of a *circulation*; intuitively, a circulation f is a flow in a (mixed) graph without sink or source nodes, i.e. where the conservation condition holds at *every* node. Formally,

Definition 2.2: A *flow function* f is a function which assigns a real number to each *ordered* pair of nodes i,j in a mixed graph such that f(i,j) is non-negative if $<i,j>$ is a directed edge from i to j, f(i,j) is zero if there is no edge between i and j, and for every pair of nodes i,j $f(i,j) = -f(j,i)$.

The interpretation of f is that f(i,j) is positive if flow moves from node i to node j, and negative if flow moves from node j to node i. In the case that (i,j) is an undirected edge, this definition allows f(i,j) (the flow *from* i *to* j) to be either positive, negative or zero, but in any case, $f(i,j) = -f(j,i)$.

Definition 2.3: The flow function f is a circulation if and only if $\sum_j f(i,j) = 0$ for every node i, i.e. the flow into i equals the flow out of i.

Definition 2.4: If f and g are two circulations then the *addition* of f and g is the function s(i,j)

$= f(i,j) + g(i,j)$. Note that by the use of negative f and g values, flows in opposing directions on an undirected edge will (partially) cancel, so that the magnitude of s(i,j) may be smaller than that of either f(i,j) or g(i,j). The following facts are well known and easy to prove.

Fact 1: The addition of two circulations is a circulation.

Fact 2: Given a circulation f, let X be the set of undirected edges (i,j) such that $f(i,j) > 0$, and let Y be the set of directed edges such that $f(i,j) > 0$. If each edge of X is directed from i to j, then every edge in $X \cup Y$ is contained in some simple directed cycle in $X \cup Y$.

Now we can prove Theorem 1.

Proof of Theorem 1: We start with an optimal solution to the mixed augmentation problem, and we will find a way to orient the undirected edges of G so that every edge is in some directed simple cycle in the optimal solution; this orientation defines G′ and proves the theorem.

If each edge is traversable, then there exists a set of traversable cycles that contains every edge of G. Let S be such a set of cycles extracted from the optimal solution to the mixed problem. If each undirected edge is used in only one direction in S, then we direct the undirected edges in the direction used in S, and the theorem is proved. If this is not the case, we will use S to find a set of traversable simple cycles S′ with the above property. To begin, order the cycles in S arbitrarily, and call them C(1), C(2), ..., C(k).

Let f_C be a circulation associated with cycle C and defined by the direction of traversal of C (for a cycle C composed only of undirected edges, pick a direction of traversal arbitrarily). In particular, $f_C(i,j) = 1$ if the traversal of C moves from i to j along an edge (directed or undirected) between i and j, and $f_C(i,j) = 0$ if i and j are not connected by an edge on C. To get the idea of the proof let F be the addition of circulations $f_{C(1)}$ through $f_{C(k)}$. By fact 1, F is a circulation and by fact 2, the undirected edges with non-zero flow can be directed so that all the

non-zero edges in F are contained in a set of simple directed cycles. If every edge of G has non-zero flow in F, then we are done. Although no directed edge of G will have zero flow in F, an undirected edge will have zero flow if the total flows in opposite directions exactly cancel. To overcome this problem, we will use S to find a circulation in which every edge of G has a non-zero flow. Given a circulation f, define 2f to be the addition of f with itself. Circulation F(k) is constructed as follows.

1) set F(0) to the circulation F(i,j) = 0 for every ordered pair i,j.

2) For i = 1 through k set F(i) to $2F(i\text{-}1) + f_{C(i)}$.

We claim that F(i) is a circulation with non-zero flow on every edge in the union of edges in cycles C(1) through C(i). This follows easily by induction on i. Clearly it is true for i = 1. Now consider $F(2) = 2F(1) + f_{C(2)} = 2f_{C(1)} + f_{C(2)}$. Let (i,j) be an edge in $C(1) \cap C(2)$. If the direction of flow on (i,j) is the same in circulations $f_{C(1)}$ and $f_{C(2)}$, then in F(2), edge (i,j) has flow of value three in that same direction. If the direction of flow on (i,j) in $f_{C(1)}$ is opposite to the direction of flow in $f_{C(2)}$, then in F(2) there is one unit of flow in the direction of flow on (i,j) in $f_{C(1)}$. Clearly for any other edge in $C(1) \cup C(2)$, the flow in F(2) is at least one. Intuitively, the purpose of adding $f_{C(1)}$ to itself before addition to $f_{C(2)}$ is to guarantee that in the resulting circulation every edge in $C(1) \cup C(2)$ has non-zero flow. The same argument is used in the general induction step, and hence F(k) is a circulation in which all edges of G have non-zero flow. Then, by fact 2, Theorem 1 is proved. □

Corollary 1: In F(k), the direction of flow on an undirected edge e is the direction of traversal of edge e in circulation $f_{C(i)}$, where i is the smallest index such that edge e is in cycle C(i).

As a consequence, if S′ is a set of traversable simple cycles in G we can, without loss of optimality, direct the undirected edges in S′ using the above approach, and then solve the

augmentation problem on the resulting mixed graph. To see this, consider the optimal augmentation and note that S can be made to contain S′, and these cycles can be given the smallest indexes in the enumeration of the cycles of S. This observation will be used later.

Corollary 2: The undirected augmentation problem for connected graphs treated in [ET] is equivalent to the undirected problem defined in this paper, i.e. there is no loss of optimality in adding directed edges to an connected undirected graph so that every edge is traversable, instead of adding undirected edges so that every edge is in a cycle.

Corollary 3: A connected mixed graph is traversable if and only if its undirected edges can be directed so that the resulting directed graph is strongly connected. This follows from Theorem 1 and Fact 0.

2.1.1. The communication completion problem

Corollary 3 implies that the communication completion problem is equivalent to the mixed augmentation problem. Recall that in the communication completion problem we add the fewest directed edges and direct undirected edges so that the resulting directed graph is strongly connected. In the mixed augmentation problem we add the fewest directed edges so that the resulting mixed graph is traversable, and hence, by Corollary 3, it can be directed to become strongly connected. So a solution to the mixed augmentation problem is a solution to the communication completion problem. Conversely, any solution to the communication completion problem is clearly a solution to the mixed graph augmentation problem, since any directed edge in a strong component is in a directed cycle.

2.2. The Orientation Problem: use of CG

Given Theorem 1, we can solve the mixed augmentation problem for G if we can efficiently find the orientation of the undirected edges of G which leads to the best solution of the resulting directed augmentation problem. Fortunately, the cost of the optimal solution for the directed augmentation problem (when, as assumed, $\bar{G}$ is a complete directed graph) has a simple form. We first need some definitions.

Definition 2.5: In a mixed graph G we call a node an *in-node* if all edges incident with it are directed into it, and call it an *out-node* if all incident edges are directed out of it; the remaining nodes of degree at least two are called *through* nodes.

Definition 2.6: Let I(G) and O(G) be respectively the number in and out-nodes of mixed graph G.

Definition 2.7: Let H be a subgraph of G = (V,E). The graph resulting from *condensing* H *in* G is a graph with node set V plus new node h, minus the nodes of H; all edges in G with both endpoints in H are removed; for any node u in H and v not in H, the edge (u,v) is replaced by (h,v), and any directed edges <u,v> or <v,u> are similarly replaced by <h,v> or <v,h>.

Definition 2.8: For a directed graph G′, let SG′ be the acyclic directed graph resulting from condensing each strong component in G′ to a single node.

It is shown in [ET] that M(G′) = max[I(SG′), O(SG′)]. Hence the mixed augmentation problem reduces to the problem of orienting the undirected edges of G, forming directed graph G′, in a way that minimizes max[I(SG′), O(SG′)]. What makes this problem difficult at this point is the indirect way (due to the intermediate step of condensing G′ to SG′) that choices about orienting undirected edges of G affect the number of in and out-nodes in the resulting SG′. To attack the problem more directly, we would like to reverse the order of the operations, i.e.

condense first and then orient. We can do this using graph CG.

2.2.1. Graph CG

We construct the mixed graph CG as follows:

Construction of CG

1. Find the strongly connected components of G, interpreting any undirected edge in G as two directed edges, one in each direction. Let A be the set of edges not in any of these strong components. Clearly, A consists only of directed edges.

2. Consider each of the strong components as a separate undirected graph, ignoring the directions on any directed edges. Let B be the set of bridges in these components; clearly B consists only of undirected edges, since any directed edge in one of the strong components is in a cycle inside the strong component itself.

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

Figure 1 shows an example construction of CG from G. Note that it is easy to combine steps 1 and 2 into a single algorithm, using roughly half the time needed for the two separate steps. We leave this to the reader.

We make the following claims:

Lemma 1: For every ordered pair of nodes u,v in any K(i), there is a traversable simple path from u to v in K(i), and every edge in K(i) is in a traversable simple cycle in K(i).

Proof: Each K(i) is a subset of a strong component χ found in step 1 above, and hence there is a traversable simple path P from u to v in χ. But if P traverses an edge in B, then it must

traverse it twice, in which case P could not be simple. Hence P is strictly contained in K(i), and the first part of the lemma is proved. Now any directed edge <u,v> in K(i) is in the traversable simple cycle in K(i) consisting of edge <u,v> followed by a traversable simple path P from v to u, so the second claim for directed edges is proved. The situation is a little more complicated for an undirected edge (u,v) in K(i). Since (u,v) is not a bridge in χ, there must be a simple path P in K(i) connecting (ignoring the directions of the edges on P) node u and node v. If P is traversable, 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. Let P′ be a traversable simple path from y to x in K(i). If P′ contains (u,v), then P′ followed by <x,y> is a traversable simple cycle in K(i) containing (u,v). If P′ does not contain (u,v) then the path consisting of P to y, P′ to x, and P to v is a path which traverses in the wrong direction one less edge than does P. Hence by repeating this argument, a traversable cycle in K(i) containing (u,v) is found, and from that a traversable simple cycle containing (u,v) can be extracted. □

Lemma 2: The optimal augmentation for CG is optimal for G.

Proof: Clearly, any augmentation for G defines an augmentation for CG at equal or less cost since CG is derived from G by condensing nodes of G. So M(CG) $\leq$ M(G). Conversely, any augmentation for CG is an augmentation for G since every edge in any K(i) is in a traversable cycle in G, and further, there is a traversable path between ordered pair of nodes in K(i). Hence any augmentation of CG is an augmentation for G, and M(G) $\leq$ M(CG). □

2.2.2. Advantage of CG

We have reduced the augmentation problem on G to the augmentation problem on CG, but it isn't clear that we have made any progress. We saw earlier that the optimal augmentation of G can be found by finding a G′ to minimize max[I(SG′), O(SG′)], but that this was difficult to do because of the intermediate step of condensing G′ to SG′. This difficulty is overcome with the

use of CG; In CG we can see directly how a chosen orientation of undirected edges in G affects the number of in and out-nodes in SG′. The following assumption and lemma make this precise.

Assumption: Since every edge in any K(i) is in a traversable cycle in K(i), we can, following the discussion after corollary 1, orient the edges in each K(i) without loss of optimality. From this point on we will assume that this has been done, so that when we refer to G′ it is assumed that the orientation of the edges of K are given as determined above.

Lemma 3: Let G′ be an orientation of the undirected edges of G, and let CG′ be an orientation of the undirected edges B of CG in the same way they are directed in G′. Then SG′ = CG′.

Lemma 3 says that after orienting edges B of CG, no further condensation is needed to form SG′ (where G′ results from G by the same orientation of B), hence max[I(SG′), O(SG′)] can be computed immediately after orienting CG. This makes the problem of finding the best G′ more directly solvable.

Proof of Lemma 3: Since every edge in K(i) is in a traversable cycle in K(i), K(i) is a strong component in G′. It is a maximal strong component in G′ because no edge of A ∪ B is in a traversable cycle in CG, and hence is in no directed cycle in G′. Therefore the K(i) sets define the nodes of SG′ and the edges of SG′ are exactly A ∪ B. Hence SG′ and SG′ have the same node and edge sets, and the edges of B are directed the same way in both. □

2.3. Optimal orientation of CG

CG is a mixed acyclic graph in which the undirected edges form a forest of undirected trees. The mixed augmentation problem has been reduced to one of orienting the edges of this forest to minimize the maximum number of resulting in and out nodes (recall that some in and out-nodes will already exist in the directed parts of CG.) In this section we develop the ideas needed to

find an optimal orientation. In particular, we look closely at how the directed edges of CG touch the undirected trees of CG, and how this affects the optimal solution. Let T be one of the trees in the undirected forest of CG.

Definition 2.8: We call an edge not in T an *in-edge* if it is directed into T and an *out-edge* if it is directed out of T.

Lemma 4: Without loss of optimality, we can assume that any leaf of T is either adjacent with no directed edge or is adjacent to both an in and an out edge.

Proof: Consider a leaf v incident only with in-edges. We can orient the undirected tree edge incident with v away from v (making v a through node); this orientation preserves optimality, since if it is oriented the other way in the optimal orientation then v is an in-node and reversing the direction of the edge will not increase the number of in or out-nodes. We can continue such orientations until no leaf (in the trees formed by the edges that remain undirected) is incident only with in-edges. The analysis is the same for out-edges. □

2.3.1. Effect of edges incident on T

We assume that every tree T in CG has the form specified in Lemma 4. We now examine how the number and location of directed edges incident on a tree T in CG affects the number of in and out-nodes that can be generated by orienting the edges of T. We are interested in the number of nodes, x, of T that become in-nodes and the number, y, that become out-nodes. In particular, we want only undominated pairs (x,y) of numbers. A pair is undominated if given the number x, the number y is the smallest integer such that there is an orientation of T resulting in x in-nodes and y out-nodes. The following lemma is fundamental in this analysis.

Lemma 5: Let T be an undirected tree with n leaves. For any $x \geq 1$ and $y \geq 1$ such that $x + y = n$, and for any arbitrary set I of x leaves of T, we can orient the edges of T so that the x

leaves of I are in-nodes, the y leaves out of I are out-nodes, and all of the interior nodes of T are through nodes.

Proof: Arbitrarily select a node v in I, and let T′ be the unique smallest subtree of T connecting v and the y leaves of T not in I (of course T′ might have some other interior nodes). Considering v as the root of T′ orient the edges of T′ so that there is a directed path from each leaf in T′ to v. After this orientation all of the y leaves of T not in I become out-nodes, v becomes an in-node, and every interior node of T′ becomes a through node since it lies on one of the directed paths created by the orientation. The remaining undirected edges of T form subtrees where in each such tree exactly one of the leaves, x, is an interior node of T′, and so x is already a through node; all the other leaves are in I, and the remaining nodes are interior nodes of T. In each of these trees, orient the edges so that there is a directed path from x to each of the leaves. All of the interior nodes in each tree then become through nodes, and all the nodes of I become in-nodes. □

Note, there are other simple schemes to orient T that prove Lemma 5. Note also that it is not possible to create fewer than n in and out-nodes in T, since each leaf becomes either an in or an out-node. Also, in any orientation at least one in-node and at least one out - node are created. Suppose not, and consider an orientation of T in which all leaves are out-nodes. Pick one of these out-nodes, v, arbitrarily, and walk along any directed path from v until a node w is reached that cannot be exited. Such a node will be reached since T has no cycles. Clearly w is not a leaf of T since all leaves are out-nodes and therefore cannot be reached by a directed path from v; hence w is an interior node and since it cannot be exited, it must be an in-node.

The following Corollary is easy to establish.

Corollary 4: Let T be as in Lemma 5, except that some of its interior nodes may be incident

with directed edges. If one or more interior nodes are incident with an in-edge, then (n,0) is added to the set of feasible undominated pairs, and if one or more interior nodes are incident with an out-edge, then (0,n) is added. No other undominated pairs are possible.

Corollary 5: If T is a tree with n leaves, and $k > 0$ of them are incident with a directed edge (from Lemma 4 they are all incident with in-edges or all incident with out-edges), then the undominated pairs x and y are all the integer solutions to $x + y = n - k$, $x \geq 0$, $y \geq 0$.

Proof: The cases where x and y are both at least one follow immediately from Lemma 5. To achieve $x = 0$ ($y = 0$), choose a node v which is incident with an out-edge (in-edge), orient all other nodes of T towards (away from) v. Notice that when $k > 0$, the existence or non-existence of interior nodes incident with directed edges has no effect on what the undominated pairs are. □

2.3.2. Using knapsack to optimaly orient CG

Given Lemma 5 and its corollaries, we can now solve the orientation problem on CG. After finding the undominated pairs (x,y) for each tree (which can be done in linear time in the number of nodes of the trees), we solve a knapsack like problem to minimize the maximum number of resulting in and out-nodes. Formally, let x(j) and y(j) be variables associated with the j'th tree of CG, and recall that I(CG) and O(CG) are the number of in and out-nodes that CG initially contains (recall that some edges of CG are directed). Then the orientation problem is to minimize $\max[I(CG) + \sum_j [x(j)], O(CG) + \sum_j [y(j)]]$, where each x(j), y(j) pair is a feasible undominated pair for the j'th tree.

For a simpler formulation of the problem, suppose, wlog, that $I(CG) \leq O(CG)$ and define d as O(CG) - I(CG); let d(j) be called a *feasible difference* for the j'th tree T if and only if x(j) - y(j) = d(j) and (x(j), y(j)) is a feasible undominated pair for T. Then the orientation problem is to

choose feasible differences to make Σ d(j) as close to d as possible. This is a knapsack problem, but, unlike the general knapsack problem, a simple solution is known: If n(j) denotes the number of leaves of the j'th tree that are not incident with directed edges in CG, then $d(j) \leq n(j)$, and Σ d(j) is bounded by the number of nodes of CG; hence this knapsack problem can be solved in polynomial time. In particular, if there are n nodes in CG, dynamic programming solves the problem in $O(n^2)$ time. However, this can be improved to linear time in the number of nodes.

Let d(j)* be the maximum value that d(j) can take on. If $\sum_j d(j)^* \leq d$, then the optimal solution is obvious. If not, then we have the following

Theorem 2: Suppose $\sum_j d(j)^* > d$. If $\sum_j d(j)^*$ and d have the same parity, then values of d(j) can be selected to exactly add to d, and if they have different parity, then values can be found to add to d+1, and this is optimal. In fact, the optimal values can be found in a greedy manner.

The method and proof require a close examination of the possible values that each d(j) can take on. The following fact is easy to establish.

Fact 3: If n(j) is odd then both 1 and -1 are feasible values for d(j); if n(j) is even then 0 is a feasible value for d(j); if n(j) is even (odd) then d(j) can take on all even (odd) values between its maximum and minimum possible value.

Proof of Theorem 2: A full proof involves several case analyses. We will examine part of the case when both d and Σ d(j)* are even; the full analysis of this case and the other cases follow in a similar manner.

Let k be the smallest integer such that $\sum_j d(j)^* < d \leq \sum_j d(j)^*$. Let I be the set of numbers d(1)*,...,d(k-1)*, and let I* be the set of all the d(j)* numbers. Since $\sum_j d(j)^*$ is even, I* must contain an even number of odd numbers, and since d is even, $d - \sum_j d(j)^*$ is even or odd depending on whether I contains an even or odd number of odd numbers. Suppose $d - \sum_j d(j)^*$ is

odd and d(k)* is also odd. Then we set d(j) = d(j)* for j = 1,...,k-1 and set d(k) = d - $\sum_j$ d(j)*, which is possible by Fact 3. For any other j such that d(j)* is even, set d(j) = 0. Now the number of variables not yet set must be even, and for each, d(j)* must be odd. We pair these variables so that in each pair, one is set to 1 and one to -1. Hence $\sum_j$ d(j) = d exactly.

3. Additional Applications

We have already shown that the communication completion problem is a mixed augmentation problem, and that the mixed augmentation problem is a proper generalization of the undirected and directed augmentation problems. In this section we briefly discuss some additional applications of the mixed augmentation problem and the methodology developed above. A full discussion requires details of the solutions for the directed and undirected augmentation problems presented in [ET]. We refer the reader to that paper, and omit some details and proofs in this section.

As a first application, consider the undirected augmentation problem; In this case, CG is a single undirected tree, and we can solve the undirected augmentation problem by first orienting CG, as in the proof of Lemma 5, with x = $\lceil n/2 \rceil$ and y = $\lfloor n/2 \rfloor$, and then using the algorithm in [ET] to solve the directed augmentation problem. This approach reduces the undirected problem to the directed problem, and we claim that this two step approach yields, as a special case, the algorithm in [ET] for undirected augmentation. The key point is that the solutions of both the mixed and the directed problems permit choices at several points, and the solution to the undirected problem given in [ET] can be derived by fixing particular choices. A proof of this is easy but but is omitted as it requires details of the two algorithms.

As a second application, consider the following special form of the augmentation problem that is central in problems of statistical security [G]. Graph G contains at least three nodes, is

connected, undirected and bipartite with node set R on one side of the graph, and node set C on the other. $\bar{G}$ is also bipartite with the same node sets, and for every $i \in R$ and $j \in C$, $\bar{G}$ contains the directed edges $<i,j>$ and $<j,i>$ if and only if (i,j) is not in G; $\bar{G}$ contains no other edges but these. Hence this special case is a mixed augmentation problem in which any added edge must run between an R node and a C node in G, and unlike the general mixed augmentation problem, no edge can be added between nodes adjacent in G. We call this form of the problem the *positive* problem. Details of the security problem and how it reduces to this augmentation problem are given in [G].

This problem at first seems harder than the case when $\bar{G}$ is a complete directed graph, since $\bar{G}$ here is so constrained. We will show, however, that the methodology developed can also optimally solve this problem in linear time. We will need the following fact:

Fact 4: The algorithm and implementation for the linear time solution in [ET] of the directed augmentation problem can be modified so that every added edge extends between an in-node and an out-node.

We now develop the solution to the positive problem. First, form (undirected, connected) graph CG from G as before, and give a leaf in CG a label R if it is an original R node in G, give it a label C if it is an original C node in G, and otherwise give it label B indicating that it is a condensed node and therefore contains nodes of both labels. Now make CG a mixed graph CG′ by directing any edge incident with an R leaf into the leaf, and directing any edge incident with a C leaf out of the leaf. In the resulting graph CG′, all R leaves are in-nodes, all C leaves are out-nodes, all interior nodes are through nodes, and edges incident with B leaves are undirected. Now solve the mixed augmentation problem on CG′, i.e. where node labels are ignored, and it is permitted to add any edge between two nodes in CG′. Let Z be the set of edges added in the optimal solution to the mixed problem on CG′. We will now use Z to derive an optimal solution

to the positive problem on G.

We first discuss how to obtain a solution to the positive problem, and then discuss optimality. If Z contains no edges between adjacent nodes in CG, and contains no edges that touch a B node in CG, then Z itself clearly is a solution to the positive problem, i.e. it makes every edge in G traversable and contains no illegal edges. If Z does contain an edge, e = (a,b), where a and b are adjacent nodes in CG, then, by Fact 4, one endpoint of e must be an in-node and one an out-node of CG′. But this can only happen if both endpoints of e are leaves in CG, and since G is connected and contains more than two nodes, this would imply that one of the endpoints of e is a B node in CG. We assume, wlog, that b is a B node and a is an R node in CG.

Now each B node in CG represents at least four nodes of G, so two of these nodes are C nodes in G. Further, only one node of the nodes represented by b is adjacent with node a in G (otherwise e would not be a bridge and so would not be in CG). Hence, of the nodes represented by b, there is a C node, x, which is not adjacent to node a in G, and therefore an edge between x and a is legal. We add the edge (x,a) to G with the same orientation that e has. The set of edges produced in this way clearly makes G traversable, and hence is a legal solution to the positive problem.

To show that the solution is optimal, note first that the number of edges added to G is exactly |Z|, since each edge added corresponds to an edge in Z. Let r,c, and be the number of R, C and B leaves in CG respectively. From our discussion of the mixed augmentation problem, we know that the number of edges in Z is max(r,c) if $|c - r| > b$, and otherwise is $\lceil (r+ c+b)/2 \rceil$. These are clearly lower bounds on the number of new edges needed to solve the positive problem, hence the solution is optimal.

Since the mixed augmentation problem can be solved in time linear in the number of edges in

G, the positive security problem can also be. In [G] we present a solution to the security problem that is based on the algorithm in [ET] for the undirected augmentation problem. That algorithm is slower than the method outlined here, but the entire algorithm (including the parts from [ET]) is simpler and the presentation is self-contained.

Although we have been assuming throughout this paper that G, and hence CG is connected, this will not be true in general. When CG is disconnected, there exists a way to connect the trees of CG with a set of legal, node disjoint edges between leaves, so that these edges together with the solution to the positive problem on the resulting tree, is a solution to the original positive problem. Details of this are given in [G].

3.1. Open Problem

The most general form of the security problem discussed in [G] can be expressed as follows: Graph G is a *mixed* bipartite graph with node sets R and C, where the only directed edges in G extend from a node in R to a node in C. As above, $\bar{G}$ contains only edges between nodes that are non-adjacent in G, and which run between R and C. For every such non-adjacent pair of nodes (i,j) in G, $\bar{G}$ will contain the edge directed edge <i,j>, but may or may not contain edge <j,i>. So the difference between this problem and the positive problem is that G may be a mixed graph (although it contains no directed edges from C to R), and that for some non-adjacent node pairs (i,j) in G, graph $\bar{G}$ does not contain <j,i>. It is known, however, that for the class of graphs that arise in the general security problem, G and $\bar{G}$ have the property that the addition of all edges in $\bar{G}$ to G results in a traversable graph. It is an open question whether this augmentation problem is hard or not.

4. References

[ET] K.P. Eswaren, and R.E. Tarjan, Augmentation Problems. SIAM J. on Computing, Vol. 5 # 4, Dec. 1976.

[FJ] G.N. Fredrickson, and J. Ja'Ja, Approximation Algorithms for Several Graph Augmentation Problems. SIAM J. on Computing, Vol. 10 # 2, May 1981.

[G] D. Gusfield, A Graphical Approach to Data Security Problems. Department of Computer Science, Yale University Technical Report # Yale/DCS/TR-236, August 1984.

[O] O. Ore, Graphs and their Uses. Random House, 1963.

G:

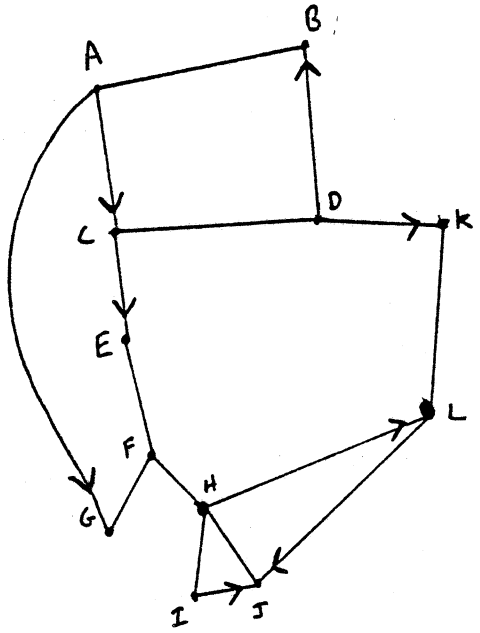


Fig. 1a
Mixed graph G

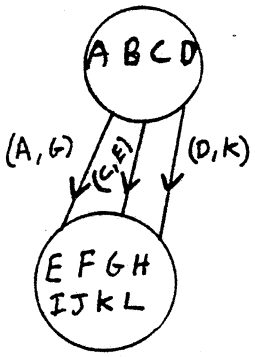


Fig. 1b
Strong components of G and the edge set A

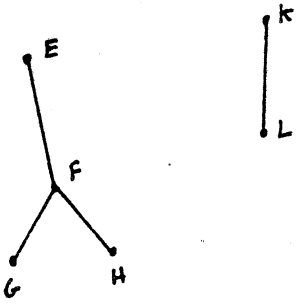
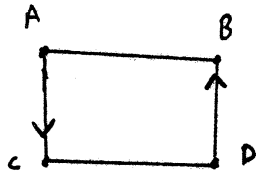


Fig. 1c
Undirected bridges inside strong component EFGH
IJKL



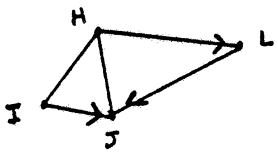


Fig. 1d
Graph K

CG:

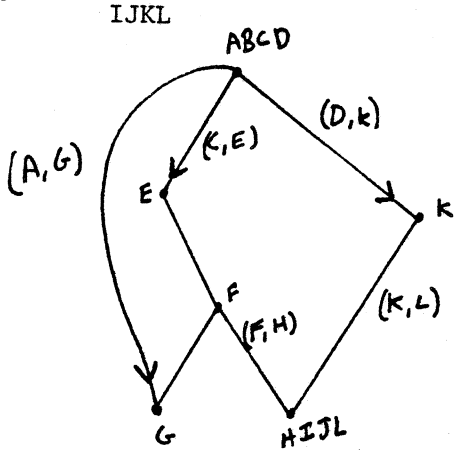


Fig. 1e
Graph CG

The Augmentation problem for G can b solved by adding one edge. There are many ways to do it. For example from K to B or from F to C.