NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN ALGORITHM FOR CLASSIFYING PSTN SWITCHING STATIONS

by

John F. Brandeau

September 1998

Thesis Advisor:	Robert F. Dell
Second Reader:	Norman D. Curet

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1998	Master's Thesis

4. TITLE AND SUBTITLE
AN ALGORITHM FOR CLASSIFYING PSTN SWITCHING STATIONS

5. FUNDING NUMBERS

6. AUTHOR(S)
Brandeau, John F.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Security Agency

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The National Security Agency (NSA) collects and processes signals intelligence information for national security purposes. As part of this mission, NSA predicts message routing over public switched telephone networks (PSTNs). The hierarchical switching level (or classification) of PSTN switching stations must be determined before making routing predictions. This thesis develops a fast graph-theoretic algorithm for accomplishing this classification. An undirected connected graph models a target PSTN; switching stations are nodes and logical connections between the switching stations are unit-length arcs. We develop bounds for the minimum number of switching levels and implicitly enumerate all possible classifications for each PSTN. The algorithm is implemented in Java and PSTNs are classified using a personal computer. Solutions are obtained in under one second for nine real-world PSTNs, and large notional networks of over 300 nodes and 900 arcs are classified in under one minute. This research improves existing node classification software.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Public Switched Telephone Network, PSTN, Hierarchical PSTN, telecommunications, Java, graph theory, National Security	66
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited

AN ALGORITHM FOR CLASSIFYING PSTN SWITCHING STATIONS

John F. Brandeau
Commander, United States Navy
B.S., Duke University, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1998

Author: _____
John F. Brandeau

Approved by: _____
Robert F. Dell, Thesis Advisor

_____
Norman D. Curet, Second Reader

_____
Richard E. Rosenthal, Chair
Operations Research Department

ABSTRACT

The National Security Agency (NSA) collects and processes signals intelligence information for national security purposes. As part of this mission, NSA predicts message routing over public switched telephone networks (PSTNs). The hierarchical switching level (or classification) of PSTN switching stations must be determined before making routing predictions. This thesis develops a fast graph-theoretic algorithm for accomplishing this classification. An undirected connected graph models a target PSTN; switching stations are nodes and logical connections between the switching stations are unit-length arcs. We develop bounds for the minimum number of switching levels and implicitly enumerate all possible classifications for each PSTN. The algorithm is implemented in Java and PSTNs are classified using a personal computer. Solutions are obtained in under one second for nine real-world PSTNs, and large notional networks of over 300 nodes and 900 arcs are classified in under one minute. This research improves existing node classification software.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The National Security Agency (NSA) conducts research and development to meet the needs of the United States for signals intelligence and communications security. Current NSA research includes development of a General Communications Assessment Tool (GCAT) to model and analyze public switched telephone networks (PSTNs). The GCAT design allows modeling of many PSTN types, where each type follows a specific switching protocol to route calls. A PSTN type used widely throughout the world is a hierarchical PSTN. In order to predict call routing in a hierarchical PSTN, its hierarchical structure must be determined (the network must be classified). A GCAT node classifier assigns each switch in the PSTN to a numerical switching level, observing a set of classification rules. After all switches are assigned to switching levels, GCAT predicts communications traffic flow through the PSTN and analyzes PSTN performance.

Existing GCAT artificial intelligence software for node classification has limited ability to classify switches in hierarchical PSTNs. This thesis develops and tests a fast, robust algorithm, called the Top Down Node Classification Algorithm (TDNCA), to classify switches in hierarchical PSTNs. An abstraction of a PSTN is a connected network whose nodes represent switches, and whose arcs represent connections between switches. Given this abstraction, TDNCA applies graph-theoretic techniques to infer the hierarchical switching levels of the network. The primary objective is to find a classification with the fewest number of hierarchical switching levels, because real-world PSTNs are constructed in this manner. We develop bounds for the minimum number of levels, and implicitly enumerate all possible classifications for each network.

TDNCA observes several classification rules that reflect the engineering design of hierarchical PSTNs. In some real-world PSTNs, the actual configuration differs from the standard hierarchical design. TDNCA has the ability to use *soft inferences* to more accurately classify a PSTN. A soft inference is the probability that a switch occupies a certain hierarchical level, based on engineering characteristics of installed PSTN switches.

This thesis explores three different ranking criteria for PSTN classifications. One ranking criterion assumes no soft inferences exist. The remaining criteria count the number of soft inferences that are satisfied and apply a quadratic penalty to soft inferences that are not satisfied. We compare the results from the three ranking criteria and make recommendations for further research.

TDNCA is implemented in Java and sample PSTNs are classified using a personal computer. Solutions are obtained in under one second for actual PSTNs. Large notional networks of over 300 nodes and 900 arcs, developed to test specific aspects of the algorithm, are classified in under one minute. TDNCA is faster than existing GCAT software and can be easily re-coded into C or C++ and integrated into GCAT.

ACKNOWLEDGMENT

I am sincerely grateful for the assistance and guidance of Prof. Rob Dell, who made key suggestions that led to the development of the algorithm presented in this thesis. He was always willing to listen to me bounce ideas off him (string and styrofoam models on his office floor, among others), usually steering me back into thesis reality. In particular, he provided valuable instruction on how to write concisely and actively.

Additional thanks to Dr. Norm Curet, who first presented this problem during my six week experience tour in the NSA OR shop. He made the experience tour worthwhile and productive, providing interesting exposure to real problems and an environment favorable to solving them.

Prof. Kevin Wood taught me network theory well in his class and provided a starting point for algorithm 2 during an office discussion. Prof. Jerry Brown provided thoughtful comments which greatly improved the quality of this thesis.

I greatly appreciate the assistance of Major Al Olson, USMC, whose many discussions helped crystallize some of my ideas. He also provided many of the figures that appear in this thesis.

I. INTRODUCTION

A. PROBLEM DESCRIPTION

Executive Order 12333 (1981) tasks the National Security Agency (NSA) with "collection and processing of signals intelligence (SIGINT) information for national foreign intelligence purposes," "executing the responsibilities of the Secretary of Defense as executive agent for the communications security (INFOSEC) of the United States Government" and conducting "research and development to meet the needs of the United States for signals intelligence and communications security."

NSA studies United States and foreign communications infrastructure for purposes of exploitation and protection. Current NSA research includes development of a General Communications Assessment Tool (GCAT) to model and analyze public switched telephone networks (PSTNs). GCAT (Figure 1) has several major functions, including modeling arbitrary inter-connected and hierarchical PSTN topologies, predicting communication routes through PSTNs, and analyzing PSTN performance.

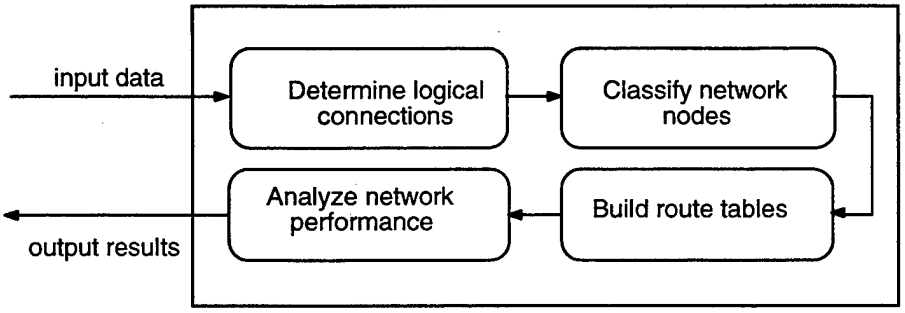


Figure 1: Schematic of Generalized Communications Assessment Tool (GCAT)

GCAT develops a network (a set of *nodes* connected by undirected, unit-length *arcs*) representative of a PSTN (a set of *switches* or *exchanges*, connected by *trunk lines* or *links*). This thesis uses the term PSTN when referring to an actual telecommunications switching structure and just the term network when referring to the modeled abstraction. A GCAT node classifier assigns each switch in the PSTN to a numerical switching level,

observing a set of classification rules. After all switches are assigned to switching levels, the route builder predicts communications traffic flow through the PSTN and analyzes PSTN performance.

GCAT design allows modeling of many PSTN types, where each PSTN type follows a specific switching protocol to route calls through the PSTN. A PSTN type used widely throughout the world is a hierarchical PSTN. Existing GCAT artificial intelligence (AI) software for node classification has limited ability to classify switches in hierarchical PSTNs. This thesis develops and tests a fast, robust algorithm, Top Down Node Classification Algorithm(TDNCA), to classify switches in hierarchical PSTNs.

B. PUBLIC SWITCHED TELEPHONE NETWORKS

1. Description and Terminology

PSTNs route telephone calls worldwide. Individual telephones connect to regional PSTNs through *local loops*. A *regional* hierarchical PSTN is sized to serve a regional area (e.g., Washington, DC). Regional PSTNs tie together with sophisticated, high capacity switches to form a global PSTN. When regional PSTNs combine, additional switching levels route traffic between the regional PSTNs. A call from one telephone to another routes through a local loop to the PSTN, then through the PSTN to another local loop which connects to the receiving telephone. The connection between the two telephones is established on demand only. When the call completes, the PSTN releases that connection and regains that calling capacity.

Figure 2 illustrates a simple hierarchical PSTN with three levels. Each *exchange* (or *switch*) in a regional hierarchical PSTN occupies one of four possible integer *switching levels*, also referred to as *classes*, ranging in value from three to six. The highest switching level has the lowest *class number*. A switch with class number three or four is a large, regional exchange known as a *transit exchange* or *tandem*. Transit

exchanges are sophisticated and more costly than switches with less capability, but have greater ability to route calls over long distances (Ash 1998).

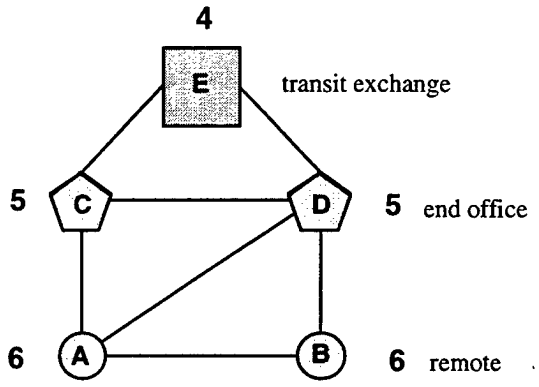


Figure 2: A simple example of a hierarchical PSTN with three levels. Letters in the circles are switch names. Numbers outside the circles are switch classes.

An exchange with class number five is an *end office*, a smaller and simpler switch than a transit exchange. Most exchanges in hierarchical PSTNs are end offices. An exchange with class number six is a *remote*, the simplest switch in a PSTN. Remotes and end offices are collectively known as *local exchanges.*

2. Logical and Physical Connections

Physical links (copper wires, fiber optic cables or microwave links), also known as trunk lines, connect PSTN switches. Actual communications between switches occur along logical connections, defined by PSTN communications software. Physical connections can be very different from logical connections, as illustrated in Figure 3. The GCAT node classifier uses logical connections.

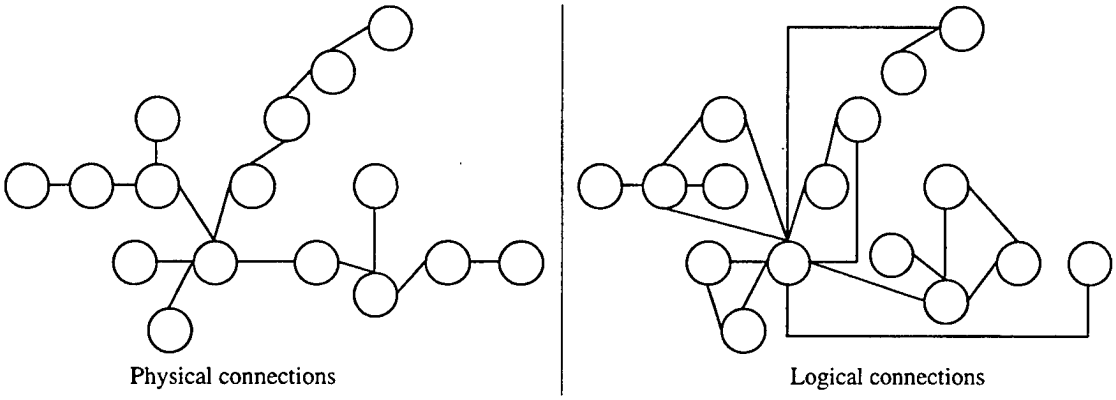
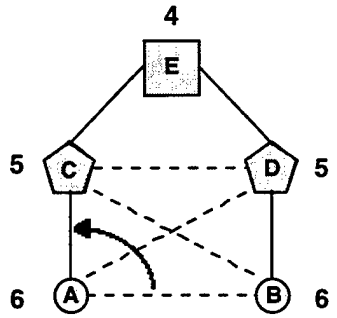


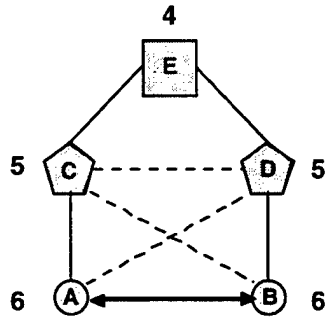
Figure 3: A simple network illustrating the difference between physical and logical connections. Communications between nodes occur along logical connections, which may traverse several physical connections. In the two diagrams, the nodes are identical, but the arcs representing connections are different.

3. Hierarchical Switching in a PSTN

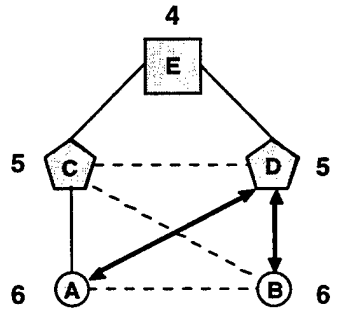
It is economically infeasible to construct PSTNs with sufficient capacity to connect all potential users simultaneously. To maximize the number of simultaneous connections supported, calls in PSTNs are routed on the most direct path available. This consumes the least switching capacity and increases the probability that a subsequent call connects. *Fixed hierarchical routing* (a characteristic of hierarchical PSTNs) is the most common architecture in use worldwide. This simple strategy was necessary several decades ago, when most PSTNs were constructed, because reliable and sophisticated switches were not available. Additionally, fixed hierarchical routing prevents switching calls back on themselves or using an excessive number of links on a call (Ash 1998). With fixed hierarchical routing, each switch must know only which switches it can communicate with directly and a priority order for connections. Each switch has exactly one *parent* (or *final*) switch (an adjacent switch closer to the top level of the hierarchy) unless the switch is itself at the top level. The parent switch is the last switch with which a lower level switch attempts to communicate. After exhausting all other switching possibilities, a switch homes on its parent switch. Figure 4 illustrates fixed hierarchical routing in a three-level PSTN.



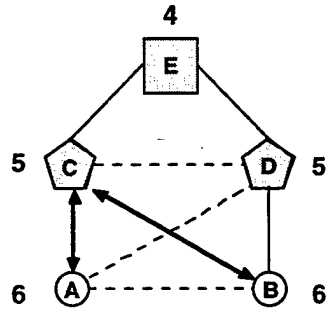
The arrow indicates the direction of homing for switch A. Switch names are inside the circles. Numbers outside the circles indicate the class of each switch.



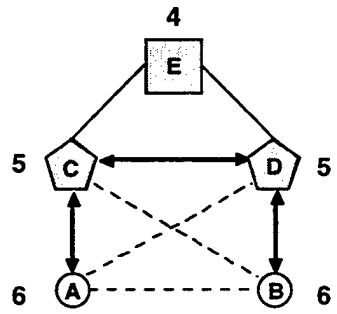
Route 1: The direct route (path A-B). This is the preferred route between A and B, using a direct link.



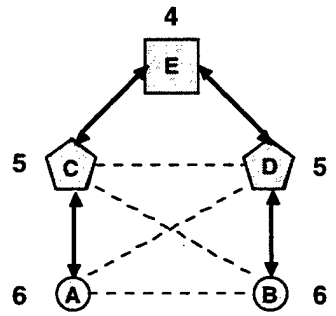
Route 2: If no direct link is available between A and B or if the link is busy, the call routes through switch D (the parent of B), then to B (path A-D-B).



Route 3: If the connection between A and D is busy or no direct link exists between A and D, the call routes through switch C (the parent of A), then to B (path A-C-B). Switch A has homed on its parent, switch C.



Route 4: If the connection between C and B is busy or no direct link exists between C and B, the call routes through both of the class five switches (path A-C-D-B).



Route 5: The final route. If route 4 is unavailable, the call routes through switch E (path A-C-E-D-B). If all circuits on the final route are full, the call is not completed.

Figure 4: Fixed hierarchical routing in a three-level PSTN. Two types of links connect switches. High–usage or direct links (dashed lines) connect switches that have sufficient traffic to make a direct route economical. Final links are the links between each switch and its parent, together with the final links between all switches at the top level (solid lines) (Ash 1998). Hierarchical routing attempts direct routes first, then overflow calls shift toward the final route. The preferred routes traverse the fewest links.

Basic structures (Figure 5) can be connected to form larger PSTNs. A mesh is a maximally-connected group of nodes, called a clique in graph theory. Each node in a mesh can communicate directly with all other nodes, allowing high traffic density. Meshes provide high reliability and are common in urban areas. A hub and spoke is a less expensive alternative to a mesh, since the switches use simpler routing. Calls route directly using links connecting adjacent outer nodes. Calls to more distant nodes route through the central node. In the star structure, all calls route through the central node. Stars are common in rural areas, where traffic density is low. Switching is most complicated in a mesh and least complicated in a star.

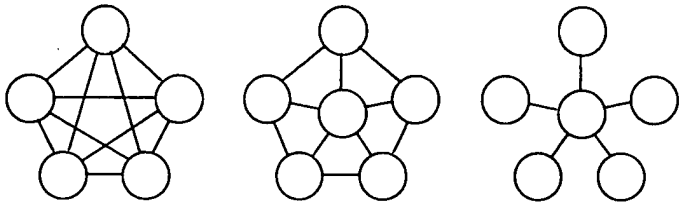


Figure 5: Typical PSTN structures. Meshes (maximally connected) are most versatile and expensive; stars (minimally connected) are simplest and least expensive. After Olson (1998)

Figure 6 illustrates a hierarchical PSTN containing each of the structures shown in Figure 5.

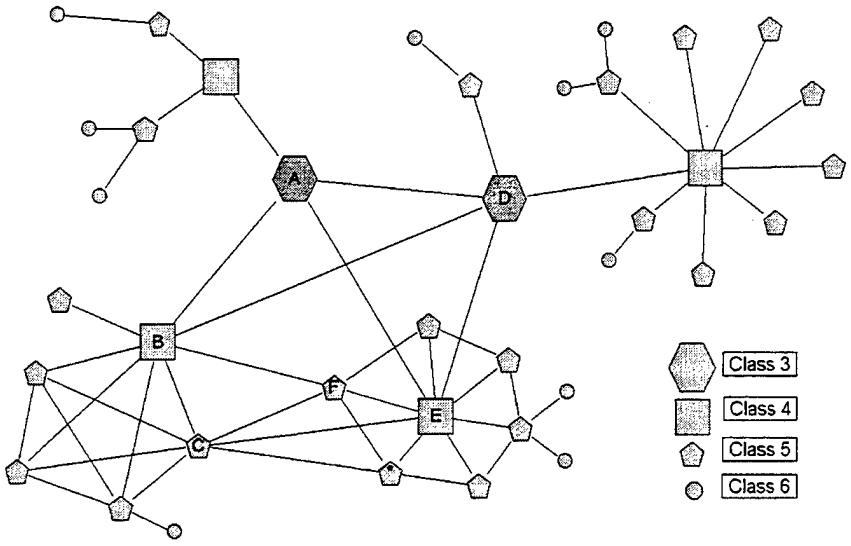


Figure 6: Schematic diagram of a typical hierarchical PSTN with four switching levels. Nodes A and D are the most sophisticated and highest capacity switches. From Olson (1998)

C. THE CLASSIFICATION PROBLEM

The class of each switch in the target PSTN is unknown. The classification problem is to "reverse engineer" the target PSTN and determine the actual switch classes. Classification rules (constraints) must be satisfied and classification objectives must be achieved.

1. Terminology

An abstraction of a PSTN is a connected network G(N, A) consisting of the set of nodes N and a set of unit-length, undirected arcs A. This thesis uses the additional terms:

1) *level* or *class number*: the number assigned to a node (integer from 3 to 6);
2) *top level*: the lowest class number assigned to a node;
3) *top-level node*: a node assigned to the top level (the highest switching level in the network);

4) *clique*: a subset of nodes $N_1 \subseteq N$ is said to be a clique if every pair of nodes in N_1 is connected by an arc (Ahuja *et al.* 1993). A clique forms a complete graph. The mesh structure of Figure 5 is an example of a clique; and

5) *solution*: an assignment of class numbers to all network nodes.

2. Classification Rules

Three hierarchical classification rules or constraints must always be satisfied:

1) top-level nodes form a clique;

2) nodes that are not top-level nodes have one parent; and

3) daughter nodes are immediate descendants of parents (each daughter is exactly one class number higher than its parent).

3. Classification Objectives

A single primary objective must always be achieved:

1) find a solution with the fewest number of levels that satisfies the classification rules.

Secondary objectives must be achieved as specified:

1) if soft inferences exist (see next section), they should be observed whenever possible, or

2) in the absence of soft inferences, the solution has a desired ratio of top-level nodes to nodes not at the top level.

Figure 7 shows an example of a simple hierarchical PSTN with a correct classification and an incorrect classification.

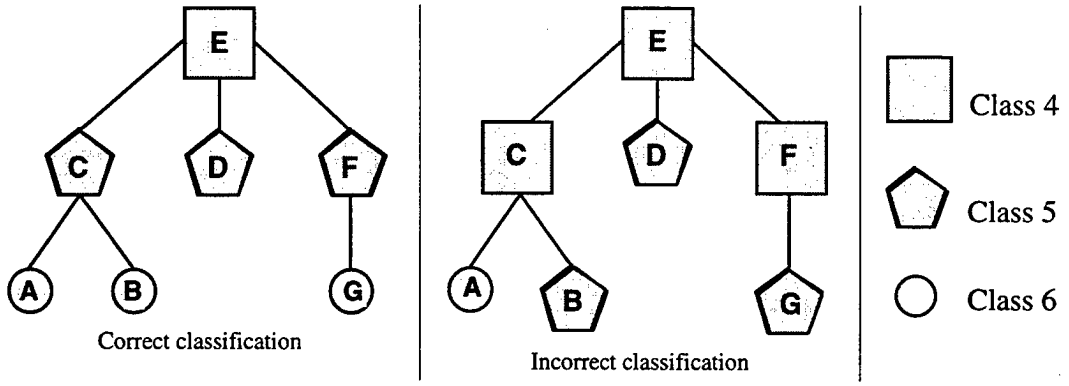


Figure 7: Examples of PSTN classifications. Arcs indicate logical connections. The correct classification follows hierarchical rules. In the incorrect classification, top-level nodes (class 4) are not a clique, since there is no arc from node C to node F.

4. Soft Inferences

Soft inferences are estimates that a PSTN switch is of a particular class, based on the engineering characteristics of the installed equipment. These engineering characteristics may be unknown, but if known, this information should be used to help classify the network. In the existing GCAT, soft inferences derive from a set of expert knowledge rules, based on equipment characteristics. In the list that follows, the rules at the top of the list are better indicators of switching level than later rules.

1) Common Language Location Identification (CLLI) rule: if the CLLI code of a switch ends in "T", this indicates a large switching capacity, normally associated with a transit exchange.

2) NPACOC rule: if the NPACOC code of a switch (which indicates the number of subscriber loops connected to the switch) is "0", the switch probably accomplishes trunk routing and is probably a transit exchange.

3) Operating Company Name (OCN) rule: if the OCN of a switch is known and is not the most common OCN of the network, the switch is most likely to be a local exchange. When multiple companies cooperatively build a PSTN or combine existing PSTNs the dominant company tends to control the transit exchanges.

4) Equipment rules: certain equipment types are more likely to be associated with certain classes of switch. Equipment types "4ESS," "DMS250," or "DMS100" are generally associated with transit exchanges. Equipment type "DCO" is generally associated with local exchanges.

5. Mathematical Formulation of the Classification Problem

A mathematical formulation of the classification problem appears below. This formulation is provided only to describe the problem.

Indices

i, j node (1, 2, 3, ... , N).

Data

LINK set of undirected arcs (i, j) in the network.

Variables

cl_i class assigned to node i;
zclass minimum class assigned to any node;
top_i 1 if node i is at the top level, 0 otherwise; and
$p_{i,j}$ 1 if node j is the parent of node i, 0 otherwise.

Formulation

There are two objectives, the former more important than the latter:

maximize *zclass* (1)

maximize $f(cl_1, cl_2, \ldots, cl_N)$ (2)

subject to:

$zclass \le cl_i \qquad \forall i$ (3)

$top_i + top_j \le 1 \qquad \forall (i, j) \notin LINK$ (4)

$$top_i + \sum_{j|(i,j)\in LINK} p_{i,j} = 1 \qquad \forall i \tag{5}$$

$$cl_i - cl_j - 4 \cdot p_{i,j} \geq -3 \qquad \forall (i,j) \in LINK \tag{6}$$

$$p_{i,j} + p_{j,i} \leq 1 \qquad \forall (i,j) \in LINK \tag{7}$$

$$cl_i - zclass + top_i \geq 1 \qquad \forall i \tag{8}$$

$$cl_i - zclass + 3 \cdot top_i \leq 3 \qquad \forall i \tag{9}$$

$$p_{i,j} \in \{0,1\} \qquad \forall (i,j) \in LINK \tag{10}$$

$$top_i \in \{0,1\} \qquad \forall i \tag{11}$$

$$cl_i \in \{3,4,5,6\} \qquad \forall i \tag{12}$$

The objective function (1) maximizes the class number of the top-level nodes, which is the same as minimizing the number of levels in the network (the primary objective). Objective function (2), an arbitrary function of the class assigned to each node, achieves the secondary objectives. Together, these objective functions achieve both classification objectives.

The constraints (3) determine the minimum class number used. This is a linearization of $zclass = \min_{i \in N} \{cl_i\}$. Constraints (4) specify that two nodes cannot both be top-level nodes if they are not directly connected by an arc. Constraints (5) require every node not at the top level to have exactly one parent node. Constraints (6) specify that if node j is the parent of node i, then node i must have a class number at least one higher than node j. Constraints (7) prevent two nodes from being parents of each other. Constraints (8) and (9) specify that if a node is not at the top level, its class number must be at least one more than the minimum class number and a node at the top level must have its class number equal to the minimum class number. Constraints (10) and (11) are binary restrictions, constraints (12) restrict variable *cl* to be integer.

D. OBJECTIVE OF CURRENT RESEARCH

The existing GCAT node classifier applies AI in the form of expert knowledge rules to determine the classification of network nodes. The rules are programmed in the C-Language Integrated Production System (CLIPS), an expert system language developed at NASA's Johnson Space Center to apply computer speed to rule-based decision making (Giarratano and Riley 1993). The AI approach to node classification has several drawbacks. It is slow to converge to a solution and is not analytically accessible (it is essentially a "black box") (Curet 1997).

The objective of this research is to develop a fast node classifier for regional hierarchical PSTNs. Inputs to the node classifier are the logical connections between network nodes as undirected, unit-length arcs, and soft inferences (if they exist). The node classifier outputs a list of nodes with assigned class numbers. This thesis develops TDNCA using graph-theoretic techniques and implements it in Java.

II. RELATED RESEARCH

Operations Research methods have contributed directly to the design and expansion of telephone networks, however nothing in the published literature directly applies to the node classification problem. This specialized problem has been researched solely by NSA.

To improve GCAT node classification speed and accuracy, Curet *et al.* (1998) developed a mixed integer linear program (MIP) at the NSA Center for Operations Research. Olson (1998) continued development of this MIP as a thesis at Naval Postgraduate School using the General Algebraic Modeling System (GAMS 1997) to generate the model and Optimization Subroutine Library (IBM 1998) to solve it.

Olson modified the formulation shown on page 10 in several ways. He uses a single objective function of the form

maximize:

$$ZWT \cdot zclass - TWT \cdot \sum_i top_i + PWT \cdot \sum_i bcl_{5i} + \sum_i \sum_c SOFT_{ci} \cdot (bcl_{ci}) \qquad \text{(obj)}$$

changes the variable cl_i to a binary variable,

bcl_{ci} a binary variable which is 1 if node i's class is c, and is 0 otherwise

and adds the following parameters as weights for the objective function,

ZWT	objective coefficient weight for minimum node class used;
TWT	objective coefficient weight for each top node;
PWT	objective coefficient weight for nodes that are not the smallest possible class given a choice of two classes; and
$SOFT_{ci}$	soft inference parameter; an objective function weight applied to influence the class c assigned to node i in the final solution.

Objective function (obj) restates lexicographic objectives (1) and (2) as a single weighted expression. The primary objective minimizes the number of levels in the classified network. This is accomplished in (obj) by the constant, *ZWT* (typically 100), multiplied by the lowest class number in the network, *zclass*. The secondary classification objective shapes the network solution, accomplished in (obj) by the remaining terms. A constant, *TWT* (typically about 0.5), multiplied by the number of top-level nodes, minimizes the number of nodes at the top level. A constant, *PWT* (typically about 0.1), encourages most nodes to be at level five. The final term allows soft inferences to contribute to the shape of the solution.

The values of the objective function weights determine the relative importance of the competing classification objectives. *ZWT* is usually much larger than either *TWT* or *PWT*, making the primary classification objective the most important. Given a set of top-level nodes, the ratio *TWT/PWT* controls the shape of the rest of the network.

In the mathematical formulation shown on page 10, constraints (4), ensuring that top-level nodes are a clique, are "weak" constraints. These constraints do not directly check that the top-level nodes are a clique, but rather ensure that among all pairs of nodes that are not linked, at most one of the nodes in the set is at the top level. This appears to cause several of the binary and integer variables to take on fractional values in the linear programming relaxation. Olson found that analysis of the network topology identifies additional constraints that tighten the formulation of the MIP and improve solution time. TDNCA uses similar but more extensive graph-theoretic techniques.

III. TOP-DOWN NODE CLASSIFICATION ALGORITHM

A. TERMINOLOGY

An abstraction of a PSTN is a connected network G(N, A) consisting of the set of nodes N and a set of unit-length, undirected arcs A. This thesis uses the additional terms:

(1) $SP_{ij} \equiv$ the shortest path distance from node i to node j; $\forall\, i, j \in \mathrm{N}$;

(2) $mSP_i = \max_{j \in \mathrm{N}} \{SP_{ij}\}$, the maximum shortest path distance from node i to any other node in the network;

(3) $maxSP = \max_{i \in \mathrm{N}} \{mSP_i\}$, the maximum shortest path distance between any two nodes in the network;

(4) $minSP = \min_{i \in \mathrm{N}} \{mSP_i\}$, mSP_i is the maximum shortest path distance from node i to any other node. *minSP* is the minimum of these maxima and, shown below in Property 4, is equal to $\left\lceil \frac{maxSP}{2} \right\rceil$;

(5) *feasible solution* ≡ a solution that satisfies all classification rules;

(6) *acceptable solution* ≡ a feasible solution that achieves the primary classification objective;

(7) *solution pool* ≡ the set of all acceptable solutions; and

(8) *optimal solution* ≡ a solution from the solution pool that best achieves the secondary classification objectives.

B. INTRODUCTION

TDNCA classifies network nodes using a "top down" approach, finding all combinations of potential top-level nodes that produce acceptable solutions. Simple addition determines the solution for each set of potential top-level nodes. TDNCA selects the solution which most favorably achieves the secondary objectives as the

optimal solution. To illustrate the principles of TDNCA, refer to the small example network of Figure 8. Consider two cases of choosing a single top-level node.

In the first case, classify node D as a top-level node, assigned class number four. Then, assign nodes C, E, F and G class number five, since they are adjacent to node D. Assign the remaining nodes, separated from node D by two arcs, class number six.

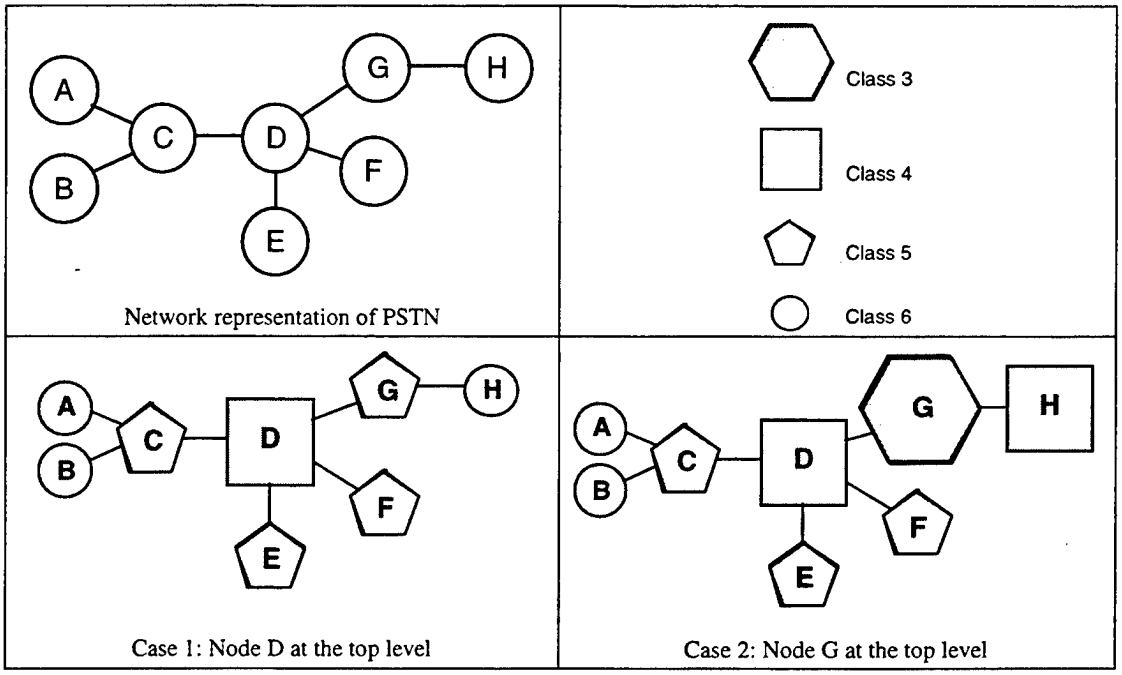


Figure 8: Example network showing two possible classifications with a single node at the top level. Assigning node D to the top level achieves the primary objective.

In the second case, classify node G as a top-level node, assigned class number three. Assign nodes D and H, adjacent to node G, class number four. Assign nodes C, E, and F, separated by two arcs from node G, class number five. Assign the remaining nodes, separated by three arcs from node G, class number six. Note that node G must be assigned class number three if it is the only top-level node so that all node class numbers are six or lower.

Of these two cases, selection of node D as a single top-level node is preferable, since this achieves the primary classification objective: a solution with the minimum

number of levels (three). If node G is selected as the single top-level node, the solution has four levels. Note that a solution follows quickly by simple addition after identifying top-level nodes. In this simple example, node D is the only single node that can be at the top level and result in a solution with three levels.

A physical analogy illustrates the idea of TDNCA. Visualize the network as marbles (nodes) connected by unit-length string segments (arcs). Lay the entire network on a floor, in two dimensions. Select a set of top-level marbles that form a clique. Tape them together as a single top-level marble. Pick up the top-level marbles and raise them until all marbles are off the floor. Then lower the top-level marbles until a marble touches the floor. Assign the marbles on the floor a class number of six. Assign the marbles on each successive higher level one class number lower than the level below. When the top-level marbles are assigned a class number, classification is complete.

As an example, select node (marble) D in Figure 8 as the single top-level marble. Raise marble D two levels (to level four). Marbles C, E, F and G occupy level five and marbles A, B and H remain on the floor (level six). Return all marbles to the floor and select marble G as the top-level marble. Marble G must be raised three levels (to level three) to classify the network. These are the results tabulated in Figure 8.

Still considering the network of Figure 8, the minimum number of levels for any feasible solution is three. It is possible to have more than one top-level node as long as all top-level nodes form a clique, but this does not produce a feasible solution with two levels. Selecting node C along with node D raises nodes A and B to level five, but node H remains at level six. Selecting node G along with node D raises node H to level five, but nodes A and B remain at level six.

TDNCA finds all sets of top-level nodes that result in acceptable solutions. In the simple example above, the choice is obvious. For a large PSTN, the solution may not be as obvious. TDNCA uses *implicit enumeration*; it limits potential top-level nodes to the subset of nodes capable of being at the top level of an acceptable solution and then

finds all cliques in the set of potential top-level nodes. The following section establishes the mathematical basis for finding potential top-level nodes.

C. MATHEMATICAL BASIS

Six properties form the mathematical basis for TDNCA. Properties 1 and 2 bound the number of levels in an acceptable solution. Property 3 shows that multiple top-level nodes will not decrease *maxSP* by more than one. Property 4 establishes the relationship between *minSP* and *maxSP*. Properties 5 and 6 establish achievable lower limits for the number of levels in a feasible solution.

1. Upper Limit on Number of Levels

Property 1: It is always possible to classify a network with *minSP* + 1 levels.

Proof: Let $k = \arg\min_{i \in N}\{mSP_i\}$. Then $cl_k = 6 - minSP$ and $cl_i = cl_k + SP_{ki}$, $\forall i \in N$ is a feasible solution with *minSP* + 1 levels.

2. Lower Limit on Number of Levels

Property 2: It is not possible to classify a network with less than *minSP* levels.

Proof: Let TP be any set of top-level nodes in a feasible solution and let $L = \max_{j \in N}\{\min_{i \in TP}\{SP_{ij}\}\}$. Then $cl_k = 6 - L$, $\forall k \in$ TP and $cl_i = 6 - L + \min_{k \in TP}\{SP_{ki}\}$, $\forall i \in N \setminus TP$ is a feasible solution with $L + 1$ levels (the fewest number of levels for the set TP). Since all top-level nodes form a clique and exactly one arc separates any two nodes in a clique, for any $k \in$ TP, $SP_{kj} - 1 \le \min_{i \in TP}\{SP_{ij}\}, \forall j \in N$. Therefore,

$\max_{j \in N}\{SP_{kj}\} - 1 \le \max_{j \in N}\{\min_{i \in TP}\{SP_{ij}\}\} = L$. Since $minSP - 1 \le \max_{j \in N}\{SP_{kj}\} - 1 \le L$, $minSP \le L + 1$.

3. Multiple Top-Level Nodes Decrease the Number of Levels by at Most One

Let TP be a set of top-level nodes in a feasible solution and let G′ = (N′, A′) be a revised network formed by combining all nodes in TP into a single node. Thus, N′ = N\TP ∪ {|N| +1} and A′ = {(i, j)∈A | $i, j \notin$TP} ∪ {(|N| +1, j) | (i, j)∈A, i∈TP, j∈N\TP}.

Property 3: SP_{ij} of G′ ($SP_{ij}^{G'}$) is greater than or equal to SP_{ij} of G (SP_{ij}^{G}) minus 1 ($SP_{ij}^{G'} \geq SP_{ij}^{G} - 1$) for (i, j)∈A′.

Proof: All nodes in TP must form a clique. Therefore, SP_{ij}^{G} can be at most one greater than $SP_{ij}^{G'}$.

4. Relationship Between *minSP* and *maxSP*

Property 4: $minSP = \left\lceil \frac{maxSP}{2} \right\rceil$.

Proof: *maxSP* must be even or odd. For the case of *maxSP* odd, there exists a node k such that $SP_{ik} + 1 = SP_{kj} = \left\lceil \frac{maxSP}{2} \right\rceil$, where $SP_{ij} = maxSP$. Assume $SP_{kj} \neq minSP$. This implies that either $minSP < SP_{kj}$ or $minSP > SP_{kj}$. If $minSP < SP_{kj}$, then there exists a node $j' \neq k$ such that $SP_{j'i'} < SP_{kj} \forall i' \in$ N. This implies that $SP_{ij'} + SP_{j'j} < SP_{ik} + SP_{kj}$, since $SP_{j'j} < SP_{kj}$ and $SP_{ij'} < SP_{kj} = SP_{ik} + 1$ ($SP_{ij'} \leq SP_{ik}$). This is a contradiction that $SP_{ij} = SP_{ik} + SP_{kj}$ is a shortest path.

If $minSP > SP_{kj}$, then there exists a node $j' \in$ N such that $SP_{kj'} > SP_{kj}$, implying either $SP_{ij'} = SP_{ik} + SP_{kj'} > maxSP$ or $SP_{jj'} = SP_{jk} + SP_{kj'} > maxSP$, a contradiction that $SP_{ij} = maxSP$.

For the case of *maxSP* even, there exists a node *k* such that $SP_{ik} = SP_{kj} = \frac{maxSP}{2}$, where $SP_{ij} = maxSP$ and $SP_{kj} = minSP$. The proof is identical to the odd case.

5. A Feasible Solution May Exist at the Lower Limit

Property 5: If *maxSP* is even, no feasible solution exists with *minSP* levels.

Proof: From the definitions of *minSP* and *maxSP* there exists a node *k* such that $SP_{ik} = SP_{kj} = \frac{maxSP}{2}$, where $maxSP = SP_{ij}$. By Property 4, $SP_{ik} = minSP$. By contradiction, assume we can form a solution with *minSP* levels. This implies that $\min_{j' \in TP}\{SP_{ij'}\} \leq minSP - 1$ and $\min_{j' \in TP}\{SP_{jj'}\} \leq minSP - 1$. Since $minSP - 1 = \frac{SP_{ij}}{2} - 1$, $\min_{j' \in TP}\{SP_{ij'}\} + \min_{j' \in TP}\{SP_{jj'}\} \leq SP_{ij} - 2$, contradicting Property 3 since this implies $SP_{ij}^{G'} \leq SP_{ij}^{G} - 2$.

Property 6: If *maxSP* is odd, the minimum number of levels in any feasible solution is either *minSP* or *minSP* + 1.

Proof: Consider two example networks, each with *maxSP* odd. In the first example (Figure 9), the network can be classified with two levels by assigning class number five to nodes 2 and 3 and class number six to nodes 1 and 4. The minimum number of levels is equal to *minSP*.

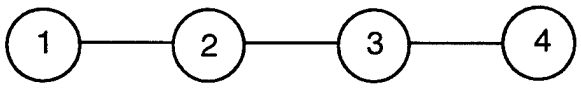
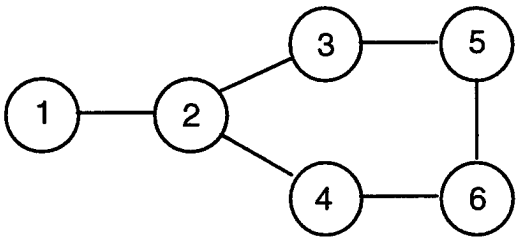


Figure 9: Example network with *maxSP* = 3, *minSP* = 2.

In the second example (Figure 10), the network cannot be classified with *minSP* levels. Table 1 enumerates all possible feasible solutions for this network. Note that

there are no cliques of three nodes in this network. The minimum number of levels is three, or *minSP* + 1.



All-pairs matrix

node	1	2	3	4	5	6
1	0	1	2	2	3	3
2	1	0	1	1	2	2
3	2	1	0	2	1	2
4	2	1	2	0	2	1
5	3	2	1	2	0	1
6	3	2	2	1	1	0

Figure 10: Example network with *maxSP* = 3, *minSP* = 2. The all-pairs shortest path matrix for this network appears to the right.

Classification with top-level nodes

node	**1**	**2**	**3**	**4**	**5**	**6**	**1, 2**	**2, 3**	**2, 4**	**3, 5**	**4, 6**	**5, 6**
1	3	5	6	6	6	6	4	5	5	6	6	6
2	4	4	5	5	5	5	4	4	4	5	5	5
3	5	5	4	6	4	5	5	4	5	4	6	4
4	5	5	6	4	5	4	5	5	4	6	4	4
5	6	6	5	6	3	4	6	5	6	4	5	3
6	6	6	6	5	4	3	6	6	5	5	4	3
levels	*4*	*3*	*3*	*3*	*4*	*4*	*3*	*3*	*3*	*3*	*3*	*4*

Table 1: All possible feasible solutions for the example network of Figure 10. Each column is a feasible solution. Column headings indicate the top-level nodes in each solution. The numbers in each column indicate the class number of the node in that row. The bottom row shows the number of levels in each solution.

6. Summary

If *maxSP* is even, we restrict our search to find feasible solutions with *minSP* +1 levels, since we know at least one feasible solution with *minSP* +1 levels exists.

If *maxSP* is odd, we initially restrict our search to find feasible solutions with *minSP* levels; one may exist. If no solutions are found with *minSP* levels, we look for feasible solutions with *minSP* + 1 levels and are guaranteed to find at least one.

IV. ALGORITHM DESCRIPTION

TDNCA finds all cliques of top-level nodes that produce acceptable solutions (feasible solutions with the minimum number of levels), then finds the acceptable solution that best achieves the secondary classification objectives. TDNCA has five major steps. The first step calculates the all-pairs shortest path matrix and determines the minimum number of levels in a feasible solution. Step two finds all nodes that can be at the top level in an acceptable solution. Step three finds each clique of top-level nodes that results in an acceptable solution. Step four classifies the network for each clique of top-level nodes found in step three, and step five finds the solution from this group that best satisfies the secondary classification objectives.

A. FIND SHORTEST PATHS

TDNCA finds the shortest path from each node to every other node (SP_{ij}) using a repeated all-pairs shortest path algorithm. Since all arcs are of unit length, breadth first search (BFS) solves the shortest path problem for each node in the network. All subsequent calculations use the all-pairs solution.

Below is the pseudo code for the repeated all-pairs algorithm, which has complexity O(N·S(BFS)), where S(BFS) is the time required to solve BFS. The complexity of BFS is O(A); the resulting complexity of all-pairs is O(N·A) (Ahuja *et al.* 1993, pp. *79, 145*).

```
algorithm 1: all-pairs shortest path
Input: undirected connected network G = (N, A), all arcs of unit length;
Output: N x N matrix of the shortest path distances, SP_si, ∀(s, i) ∈ N
{
    for each node s ∈ N {
        SP_si = ∞ ∀i ∈ N;
        SP_ss = 0;
        put node s onto FIFO Queue;
        while (Queue not empty) {
            pop node i from Queue;
            for (each arc (i, j) ∈ A(i),
                    where A(i) is the set of arcs incident to node i) {
                if (SP_sj = ∞) {
                    SP_sj = SP_si + 1;
                    push node j onto Queue;
                }
            }
        }
        store SP_si  ∀i ∈ N;
    }
}
```

B. IDENTIFY POTENTIAL TOP-LEVEL NODES

TDNCA identifies a set $P \subseteq N$ of potential top-level nodes, containing only those nodes that can be at the top level of an acceptable solution. Due to the clique constraint for top-level nodes, adding additional nodes to the set of top-level nodes reduces the distance from any node to the nearest top-level node by at most one (Property 3). Thus, the set of potential top-level nodes $P = \{i \in N \mid mSP_i = minSP\}$ for a solution with $minSP$ levels and $P = \{i \in N \mid minSP \leq mSP_i \leq minSP + 1\}$ for a solution with $minSP + 1$ levels. There are $2^{|P|}-1$ possible combinations of top-level nodes. Typically, |P| is much smaller than |N|.

C. ENUMERATE CLIQUES OF TOP-LEVEL NODES

Define set C as the set of all combinations of nodes in P. Each element $C_i \in C$, $i = 1, 2, \ldots, |C|$ is a unique combination of nodes. Define set $C' \subseteq C$ as the set of all cliques in C. Each element $C'_i \in C'$, $i = 1, 2, \ldots, |C'|$ is a unique clique formed from the nodes in set P. If all nodes in P form a clique, $C' = C$, otherwise $C' \subset C$. TDNCA does not necessarily enumerate the set C; it implicitly enumerates all possible combinations of the nodes in P, finding the set of cliques C′ and retaining only those cliques in C′ that result in acceptable solutions.

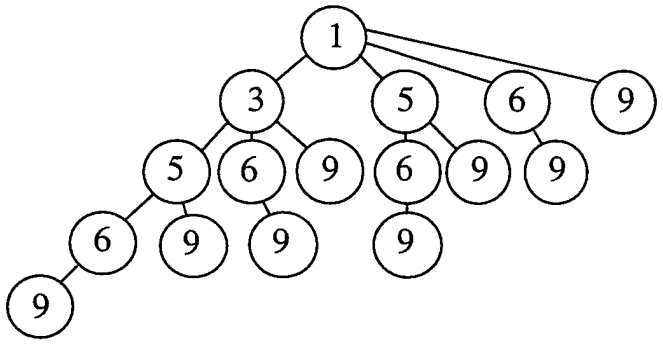
Below is the pseudo code for the enumeration of top-level node cliques. Define *ML* as one less than the minimum achievable number of levels in a feasible solution. If *maxSP* is even (Property 5) set *ML* = *minSP* (a solution with *minSP* + 1 levels). If *maxSP* is odd (Property 6) initially set *ML* = *minSP* – 1 (a solution with *minSP* levels). Since we are not guaranteed to find a solution with *minSP* levels, one is added to *ML* when necessary.

algorithm 2: enumerate cliques of top-level nodes
Input: set of potential top nodes indexed as 1, 2, 3, ... , |P|,
the all-pairs shortest path matrix and *ML*;
Output: the set of all feasible solutions with the minimum number of levels
(solution pool)

```
{
    push 1 on LIFO Stack;
    while (Stack not empty) {
        do {
            if (node at top of Stack is connected to each node on Stack) {
                if ( max_{j∈N} { min_{k∈Stack} {SP_jk} } = ML) {
                    store the Stack as a set of acceptable top nodes;
                }
                push (value on top of Stack + 1) on Stack;
            }
            else {
                pop Stack;
                if (value popped ≥ |P|) {
                    pop Stack;
                }
                push (value popped + 1) on Stack;
            }
        } until (node at top of Stack > |P|)
        pop Stack;
        pop Stack;
        if (Stack not empty) {
            pop Stack;
            if (value popped < |P|) {
                push (value popped + 1) on Stack;
            }
        }
    }
    if no acceptable solution is found, ML = ML + 1 and repeat;
}
```

Figure 11 illustrates algorithm 2 using a rooted tree constructed from the nodes in P. TDNCA explores all paths from the tree root and from each sub-root, if needed, to find all elements of set C. In Figure 11, set C = {(1), (1, 3), (1, 3, 5), (1, 3, 5, 6), (1, 3, 5, 6, 9), (1, 3, 5, 9), (1, 3, 6), (1, 3, 6, 9), (1, 3, 9), (1, 5), (1, 5, 6), (1, 5, 6, 9),

(1, 5, 9), (1, 6), (1, 6, 9), (1, 9), (3), (3, 5), ... , (9)}. Set C contains 2^5-1 = 31 elements, all possible combinations of nodes {1, 3, 5, 6, 9}.



All-pairs matrix

node	1	3	5	6	9
1	0	1	2	1	1
3	1	0	1	1	1
5	2	1	0	1	1
6	1	1	1	0	1
9	1	1	1	1	0

Figure 11: Rooted tree illustrating algorithm 2. In this example, the set P of potential top-level nodes consists of nodes 1, 3, 5, 6, and 9. The root of the tree is the lowest numbered node in set P. Construct the second level with the remaining nodes in P, each connected to the root. Starting from each node in the second level, construct sub-trees by filling the level below the root of each sub-tree with all nodes in set P numbered higher than the root of the sub-tree. Continue until all leaf nodes have the highest node number in set P. The arcs in the rooted tree do not indicate actual connections between nodes in graph G. The all-pairs shortest path matrix to the right tabulates SP_{ij} from the original graph for each node in set P. Nodes *i* and *j* are connected if $SP_{ij} = 1$, otherwise they are not connected.

Given set C, determining if each element C_i forms a clique is straightforward but computationally tedious. Algorithm 2 directly determines if each new element C_i forms a clique. If so, it continues. If not, look no farther along that branch of the rooted tree, since adding more nodes to form additional cliques is fruitless. Algorithm 2 thus finds the set C′ without explicitly finding each element of set C, significantly reducing computation in most cases.

Figure 11 shows how to find set C′, starting at the root of the tree and proceeding first down the left branch of the tree. Each added node in a path forms a new temporary set CT. If the nodes in CT are not a clique, discard CT and check no further combinations containing set CT. If the nodes in CT are a clique, store CT as the next clique C'_i, i = 1, 2, ... , |C′|. The root node (node 1) is a clique and becomes C'_1. The next set of nodes CT = (1, 3) is also a clique, so C'_2 = CT = (1, 3). Continuing down the left branch of the

tree, add node 5 to set CT. Node 5 is connected to node 3, but not to node 1 ($SP_{15} \neq 1$), so the set of nodes CT = (1, 3, 5) is not a clique. Adding additional nodes to CT to form a clique is fruitless, so we reverse direction up the tree to the last node in the previous clique, node 3. Assign to CT the contents of the previous clique. Continue adding nodes to set CT moving down the adjacent untraversed branch, keeping to the left on that branch. After adding each node to CT, check if CT is a clique. If so, save CT as the next clique in the sequence C′, $i = 1, 2, \ldots, |C'|$. If not, return to the last node in the previous clique. After traversing all branches from the root, repeat for each of the sub-trees. Subsequent cliques in the sequence are $C'_3 = (1, 3, 6)$, $C'_4 = (1, 3, 6, 9)$, $C'_5 = (1, 3, 9)$, $C'_6 = (1, 6)$, $C'_7 = (1, 6, 9)$, $C'_8 = (1, 9)$, $C'_9 = (3)$, $C'_{10} = (3, 5)$, $C'_{11} = (3, 5, 6)$, and so on. Set C′ contains 23 elements, the number of cliques in set P.

While filling the set C′, TDNCA checks that each clique $C'_i \in C'$ results in an acceptable solution. An acceptable solution for clique C'_i exists if $\max_{j \in N \setminus C'_i} \{\min_{k \in C'_i} \{SP_{jk}\}\} = ML$. If not, clique C'_i is discarded. To illustrate with Table 2, we calculate $ML = 2$. Consider two cliques of top-level nodes, $C'_1 = (G)$ and $C'_2 = (C, D)$.

mSP_i	node i	A	B	C	D	E	F	G	H
4	A	0	2	1	2	3	3	3	4
4	B	2	0	1	2	3	3	3	4
3	C	1	1	0	1	2	2	2	3
2	D	2	2	1	0	1	1	1	2
3	E	3	3	2	1	0	2	2	3
3	F	3	3	2	1	2	0	2	3
3	G	3	3	2	1	2	2	0	1
4	H	4	4	3	2	3	3	1	0

Table 2: Example all-pairs shortest path matrix for the network shown in Figure 8. For this network, *maxSP* = 4, *minSP* = 2 and *ML* = 2.

For clique C'_1, observe that $\max_{j \in N \setminus C'_1} \{\min_{k \in C'_1} \{SP_{jk}\}\} = 3$ arcs separate node G from the most distant node(s) in the network. We discard the clique C'_1 as unacceptable since $3 > ML$. For clique C'_2, $\max_{j \in N \setminus C'_2} \{\min_{k \in C'_2} \{SP_{jk}\}\} = 2$ arcs separate any node $\{j \in N \setminus C'_2\}$ from either top-

level node in clique C'_2. Since $ML = 2$, we retain clique C'_2 as acceptable. The solution with top-level node G has four levels, while the solution with top-level nodes C and D has only three levels and achieves the primary objective.

D. DEVELOP SOLUTIONS

After algorithm 2, each clique of top-level nodes remaining in set C′ produces an acceptable solution. For each clique C'_i in set C′, we classify the network by assigning each node a class number $cl_k = 6 - ML + \min_{i \in C_i}\{SP_{ik}\}, \forall k \in \mathrm{N}$. This satisfies the remaining classification rules, with each node classified exactly one level below its parent. The resulting solutions form the solution pool.

E. FIND AN OPTIMUM SOLUTION

All solutions in the solution pool satisfy the primary classification objective: classification with the fewest number of levels. TDNCA finds the solution in the solution pool that best achieves the secondary classification objectives. This thesis explores three methods for determining the "best" solution. Two methods fit the soft inferences; one uses a non-linear penalty function and the other counts the number of satisfied soft inferences. In the absence of soft inferences, the third method maximizes a linear objective function similar to that used by Olson.

1. Penalty function

Calculate a non-linear penalty for each node, a function of the node class and the class suggested by the soft inferences. Expert knowledge rules determine a soft inference for each node, expressed as the probability that the node is a transit exchange, $0 \le SOFT_i \le 1, \forall i \in \mathrm{N}$. The penalty for each node is $PEN_i = \begin{Bmatrix} (1 - SOFT_i)^2 & \text{if } cl_i \le 4 \\ SOFT_i^2 & \text{if } cl_i \ge 5 \end{Bmatrix}, \forall i \in \mathrm{N}$.

The total penalty for each pool solution is $\sum_{i=1}^{N} PEN_i$. An optimal solution by this method has the smallest total penalty.

2. Count of Soft Inference Matches

Total the matches between node class and the class suggested by soft inferences. For purposes of this method, node i is considered to be a transit exchange if $SOFT_i > 0.5$ and a local exchange if $SOFT_i < 0.5$. If $SOFT_i = 0.5$, any classification for the node matches the soft inference, so we ignore this case. The total number of matches for a solution becomes $\sum_{i=1}^{N} \begin{cases} \lfloor SOFT_i + 0.5 \rfloor & \text{if } cl_i \le 4 \\ 1 - \lfloor SOFT_i + 0.5 \rfloor & \text{if } cl_i \ge 5 \end{cases}$. An optimal solution by this method has the greatest number of matches.

3. Linear Objective function

In the absence of soft inferences, TDNCA maximizes the objective function $ZWT \cdot zclass - TWT \cdot \sum_i top_i + PWT \cdot \sum_{i \& cl_i = 5} 1$, with $ZWT = 1{,}000$, $TWT = 5$ and $PWT = 2$. Olson (1998) uses similar weights. This objective function is Olson's without his soft inference term.

V. COMPUTATIONAL RESULTS

A. HARDWARE AND SOFTWARE

TDNCA is implemented in Java (Sun Microsystems 1998). Reported results are obtained on a 333 megahertz Pentium II personal computer (PC), operating Windows NT 4.0 and the Microsoft Virtual Machine, a just-in-time Java compiler provided with Code Warrior 3.0 (Metroworks 1998).

B. TEST NETWORKS

TDNCA classifies 23 test networks, the same networks used by Olson (1998). Test networks include nine actual or modified U.S. regional PSTNs, seven notional networks constructed to test model robustness, and seven networks modified from U.S. regional PSTNs. Table 3 summarizes all test networks.

1. Networks Derived from U.S. Regional PSTNs

Nine test networks derive from U.S. regional PSTNs obtained from open sources. These networks typify regional PSTNs. Net-0 models an actual regional PSTN modified to have a leafy structure (a structure with few meshes or hubs) and four classes. Network-1 and Network-2 are variations of an actual network with a single central transit exchange. Network-3 and Network-4 are also variations of an actual network with multiple transit exchanges. Network-4 has an additional mesh attached to a transit exchange. Network-5 and Network-6 are the most complex in this group, with multiple transit exchanges and meshes. Balt and Tracy are actual regional networks. Soft inferences, based on available technical specifications, exist for all networks in this group except Net-0. Appendix A contains diagrams of the nine networks in this group.

2. Notional Networks

Olson (1998) constructs seven notional networks that are more difficult to classify. Each notional network combines several of the networks derived from U.S. regional PSTNs, tied together with additional arcs as needed. Notional networks are intended to represent real-world PSTNs, but on a larger scale.

3. Modified Networks

All but one of the test networks described above has three switching levels. To test classification of PSTNs with more than three switching levels, Olson constructs seven additional modified networks, each identified by the name "Lop", followed by the number of the original test network, then by a letter. The letter "a" indicates addition of one node to increase *maxSP* by one; the letter "b" indicates addition of two nodes increase *maxSP* by two.

Network	Location	No. of Nodes	No. of Arcs	No. of Top-Level Nodes	Max Node Degree	*maxSP*	No. of Levels in Network
Networks derived from U.S. regional PSTNs							
Net-0	Notional	21	21	1	3	6	4
Network-1	Baltimore area	27	32	1	21	4	3
Network-2	Baltimore area	38	47	1	30	4	3
Network-3	Georgia	34	38	2	10	5	3
Network-4	Georgia	34	46	3	12	5	3
Network-5	D.C. and N. VA	34	79	2	17	4	3
Network-6	D.C. and N. VA	42	96	4	16	5	3
Tracy	California	90	90	2	31	5	3
Balt	Baltimore area	103	103	3	66	5	3
Large Notional Networks							
5_6	Aggregation	76	183	6	21	5	3
4_6	Aggregation	110	247	9	24	5	3
3_6	Aggregation	144	320	12	27	5	3
HugeC	Aggregation	118	313	10	25	5	3
HugeB	Aggregation	152	402	12	27	5	3
HugeA	Aggregation	220	575	18	33	5	3
Huge	Aggregation	304	948	24	39	5	3
Networks with modified Longest-Shortest Paths							
Lop4a	Modified Network-4	35	47	3	12	6	4
Lop4b	Modified Network-4	36	48	3	12	7	4
Lop5a	Modified Network-5	35	81	3	17	5	3
Lop5b	Modified Network-5	36	82	2	17	6	4
Lop6	Modified Network-6	41	95	3	16	5	3
Lop6a	Modified Network-6	43	97	3	16	6	4
Lop6b	Modified Network-6	44	98	3	16	7	4

Table 3: Summary of test networks, adapted from Olson (1998).

C. SOFT INFERENCE VALUES

Combining expert knowledge rules to produce soft inferences is not necessarily straightforward. Each rule provides a separate indicator of the class of a switch. If several rules independently indicate that a switch is a certain type, the combined inference is stronger than any individual rule, but perhaps only slightly stronger than the strongest single rule. An inference may also be incorrect. For example, a soft inference

may be based on strong indications that a switch is a transit exchange, while in fact the switch is a local exchange. The existing GCAT AI algorithm provides individual rule weights (Curet 1997). This thesis uses estimates of combined rule weights to provide soft inferences, shown in Table 4. Soft inferences have been formulated for purposes of testing our methods, and do not have any specific technical motivation.

Applicable Expert Knowledge Rules	$SOFT_i$
OCN, DCO	0.43
OCN	0.45
OCN, DMS100	0.47
DCO	0.48
DMS100	0.52
CLLI, OCN, DCO	0.55
NPACOC, OCN	0.55
DMS250	0.57
4ESS	0.60
CLLI, OCN	0.60
NPACOC	0.60
CLLI, DMS100	0.67
CLLI	0.75
CLLI, NPACOC, OCN	0.77
CLLI, NPACOC	0.80
CLLI, OCN, DMS250	0.84
CLLI, DMS250	0.89
CLLI, 4ESS	0.95

Table 4: Soft inference values for combinations of expert knowledge rules. $SOFT_i$ = the probability that node i is a transit exchange.

Table 5 summarizes soft inference values for two real-world PSTNs. Exact values of soft inferences for each test network appear in Appendix B.

Inference Meaning	$SOFT_i$	Network Tracy Nodes	Network Balt Nodes
weak suggestion the node is not a transit exchange	< 0.5	0, 2, 6, 8, 11, 28, 29, 32, 34, 40, 43, 46, 47, 52, 55, 57, 61, 668, 71, 72, 79, 88, 89	16, 27, 48, 86, 89, 92
weak suggestion the node is a transit exchange	0.5 to 0.7	12, 19, 43, 63, 67, 76, 84, 87	2, 7, 14, 23, 52, 53, 54
strong suggestion the node is a transit exchange	> 0.7	20, 58, 80, 81	18, 48, 68, 84, 87, 102

Table 5: Soft inferences grouped into categories for two real-world PSTNs. Nodes with strong inferences are most likely to be classified as top-level nodes when considering soft inferences. Weak inferences have less influence on solutions.

D. TEST RESULTS

TDNCA classifies most test networks in less than one second and never needs more than 67 seconds. For testing purposes, TDNCA calculates three different objective functions for each network. Table 6 summarizes computational performance for all test networks.

The slower classification of network Huge results from the large number of potential top-level nodes. The number of arcs, |N|, in network Huge is 304. There are $2^{304} - 1$ possible combinations of these nodes. However, of these 304 nodes, only |P| = 24 nodes can be at the top-level of an acceptable solution. All 24 of these nodes form a clique, so there are |C| = |C'| = $2^{24} - 1$ or 16,777,215 cliques of top-level nodes. TDNCA completely enumerates these cliques to find 20,736 that result in acceptable solutions. The best solution is chosen from among this group.

Network	Number of Nodes |N|	Number of Arcs |A|	Number of feasible top-level combinations of nodes	Number of solutions in pool |C'|	Time To Classify (secs)
Net-0	21	21	7	4	0.3
Network-1	27	32	35	18	0.3
Network-2	38	47	55	28	0.3
Network-3	34	38	3	1	0.3
Network-4	34	46	7	1	0.3
Network-5	34	79	277	139	0.4
Network-6	42	96	19	8	0.3
Tracy	90	90	7	2	0.3
Balt	103	103	7	2	0.3
5_6	76	183	63	12	0.3
4_6	110	247	511	12	0.3
3_6	144	320	4095	12	0.3
HugeC	118	313	1023	72	0.4
HugeB	152	402	4095	144	0.4
HugeA	220	575	262,143	144	1.1
Huge	304	948	16.777,215	20,736	67
Lop4a	35	47	37	19	0.3
Lop4b	36	48	7	7	0.3
Lop5a	35	81	263	98	0.9
Lop5b	36	82	279	205	0.3
Lop6	41	95	289	212	0.4
Lop6a	43	97	282	206	0.4
Lop6b	44	98	19	8	0.3

Table 6: Summary of classification performance for all test networks. Calculation of the all-pairs shortest path matrix requires less than 0.1 seconds in all cases. Enumeration of feasible top-level sets requires less than 0.7 seconds for all networks except Huge, which requires 46 seconds. Time to classify includes input, output and all calculations. The test machine is a 333 megahertz Pentium II PC.

The solution pool contains all solutions that achieve the primary classification objective: classification with the minimum number of levels. This thesis ranks solutions from the solution pool using three different optimality criteria (Table 7). In all cases, at least one of the optimality criteria identifies the solution for each network that correctly classifies the network. In a correct classification, each assigned node class matches the actual class of the node in the modeled PSTN. In all but one case (network Balt), the linear objective function correctly classifies the network.

	Optimal Sets of Top-Level Nodes				
Network	Maximum Objective Function	Minimum Penalty Function		Highest Count of Soft Inference Matches	
Net-0	**13**	no soft inferences			
Network-1	**14**	**14**		*3, 14*	
Network-2	**14**	*14, 31*		*3, 14*	
Network-3	**9, 28**	**9, 28**		**9, 28**	
Network-4	**7, 9, 28**	**7, 9, 28**		**7, 9, 28**	
Network-5	**21, 31**	*0, 31*	*25, 31*	**21, 31**	*21, 24, 31*
Network-6	**6, 15, 19, 37**	**6, 15, 19, 37**		**6, 15, 19, 37**	
Tracy	**20, 81**	*20, 58, 81*		*20, 58, 81*	
Balt	*68, 84*	**68, 84, 87**		**68, 84, 87**	

Table 7: Optimal top-level node sets for test networks derived from actual U.S. PSTNs. The set of top-level nodes shown in each block best achieves the indicated objective. Split blocks indicate multiple optimal solutions. Sets in bold numbers exactly match the true configuration of the test network. Sets in italics fail to classify one actual top-level node, classify an additional top-level node, or both.

Soft inferences influence the selection of optimal solutions. In the following paragraphs, we examine the classification results for networks Tracy and Balt and illustrate the influence of soft inferences.

Classification of network Balt using the maximum objective function criterion identifies two top-level nodes, nodes 68 and 84. The actual Balt network has three top-level nodes, nodes 68, 84 and 87. For network Balt, $ML = 2$, $maxSP = 5$, and $minSP = 3$. Observe that node 87 has only three adjacent nodes (see diagram in Appendix A), $mSP_{87} = 3$ and nodes 68, 84 and 87 form a clique. The objective function minimizes the number of top-level nodes, resulting in exclusion of node 87 from the top level. Nodes 68, 84 and 87 all have strong inferences that they are transit exchanges (Table 5). Optimality functions that include soft inferences benefit from placing node 87 at the top level, since a strong soft inference is satisfied. The soft inferences in the case of network Balt contribute to a correct solution.

Classification of network Tracy using the maximum objective function criterion identifies the correct solution, but classifications using soft inferences are incorrect.

Node 58 in network Tracy is only adjacent to nodes 20 and 81 (see diagram in Appendix A). The objective function excludes node 58 from the top level to minimize the number of top-level nodes. The soft inference for node 58 strongly suggests a transit exchange, so the optimality functions that include soft inferences place node 58 at the top level. The actual Tracy network has only two top-level nodes.

A single node with few incident arcs and a strong soft inference leads to the classification errors in networks Balt and Tracy. For all test networks, the solution pool contains the correct solution and at least one of the optimality criteria identifies the correct solution.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

TDNCA quickly classifies all test networks, including two networks that model real-world PSTNs. Solutions are obtained in under one second for nine real-world PSTNs, and large notional networks of over 300 nodes and 900 arcs are classified in under one minute. The solution speed of TDNCA is due to custom algorithms written for the purpose of node classification and implemented in Java. TDNCA outperforms existing node classification software and appears fast enough to allow classification of any real-world PSTN in less than one minute.

B. COMPARISON TO PREVIOUS RESEARCH

TDNCA has several advantages over previous efforts:

1) TDNCA is written in Java, using well known graph-theoretic techniques and simple programming. It does not require the use of commercial software packages and proprietary algorithms. TDNCA can be used as written in Java or can be easily re-coded in C or C++ and integrated into GCAT.

2) TDNCA classifies test networks more quickly than previous methods and with equal accuracy. We obtain the same results as Olson for all test networks when using Olson's objective function.

3) TDNCA can evaluate multiple optimal or nearly optimal solutions in the neighborhood of the optimal solution. Once TDNCA finds the solution pool, solutions can be rank-ordered using any desired fitness function. Fitness functions need not be linear.

The advantage of the MIP used by Olson (1998) is a flexible formulation. Olson does not attempt to classify a network with a minimum number of levels, but evaluates

solutions solely based on a numerical objective. Typical MIP objective function weights result in classification with a minimum number of levels, but varying the weights can produce solutions with more than the minimum achievable number of levels.

C. RECOMMENDATIONS FOR FURTHER RESEARCH

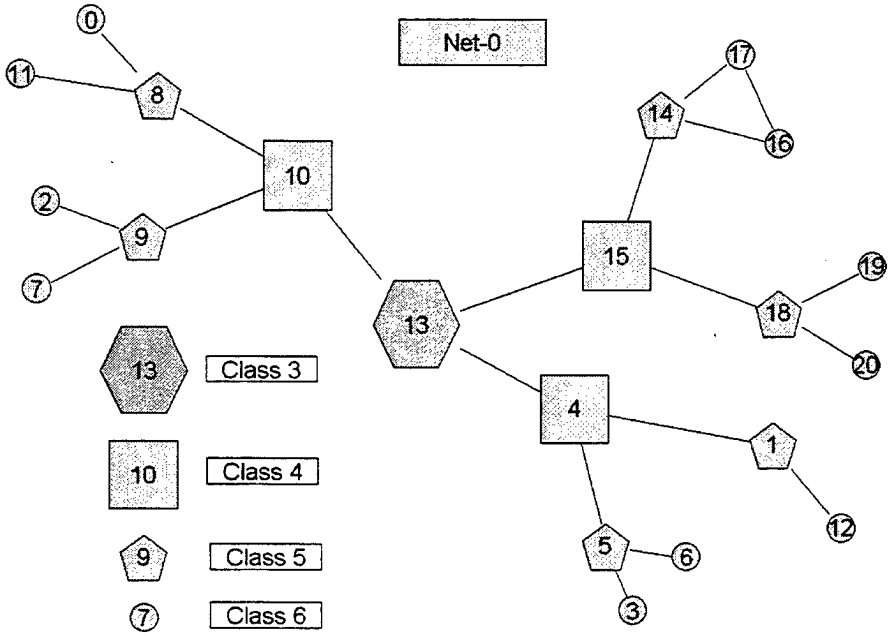
TDNCA strictly achieves the primary objective, classification of networks with the minimum number of levels. If the MIP's flexibility to classify solutions with more than the minimum number of levels is desired, TDNCA can be modified to do so.

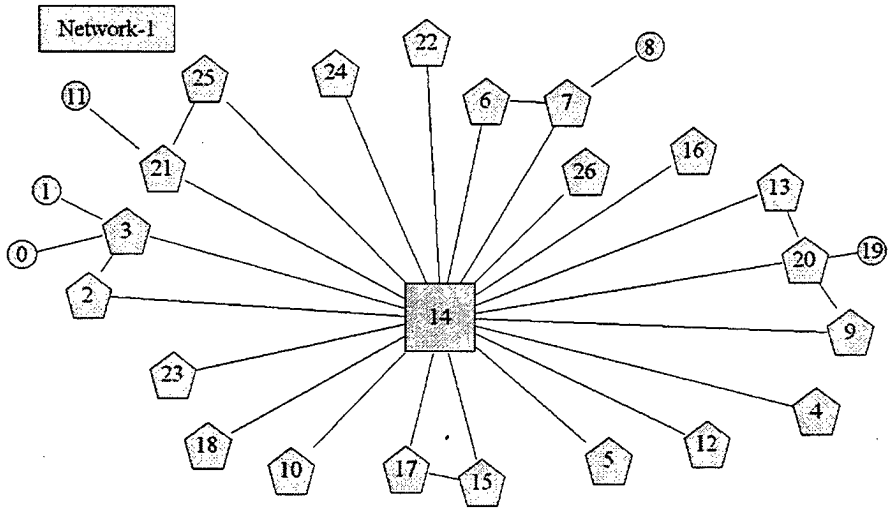
Further research is required to numerically define the "best" classification criterion. This thesis explores three methods, but no single method correctly classifies all test networks. Use of expert knowledge rules to determine soft inferences has promise, but soft inferences may lead to incorrect classification. Consistently classifying PSTNs with no errors may require a more sophisticated application of soft inferences.

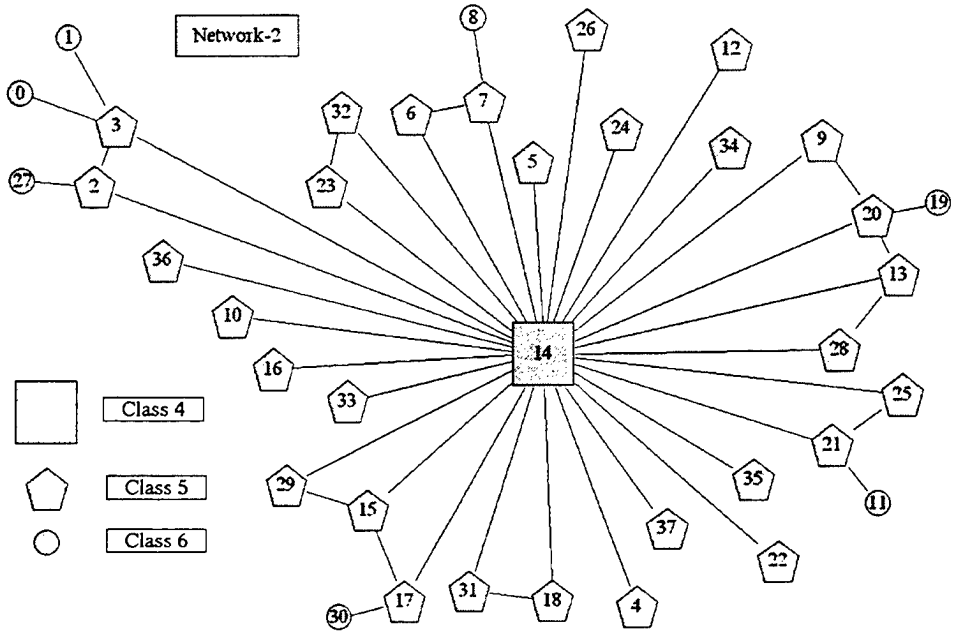
Soft inferences used in this thesis indicate the probability that a node is a transit exchange (class three or class four). TDNCA can easily be modified to use more detailed soft inferences. For example, a node might have a probability of 0.2 of being class four or higher, a probability of 0.7 of being class five and probability of 0.1 of being class six. More detailed soft inferences such as these may help to correctly classify real-world PSTNs.

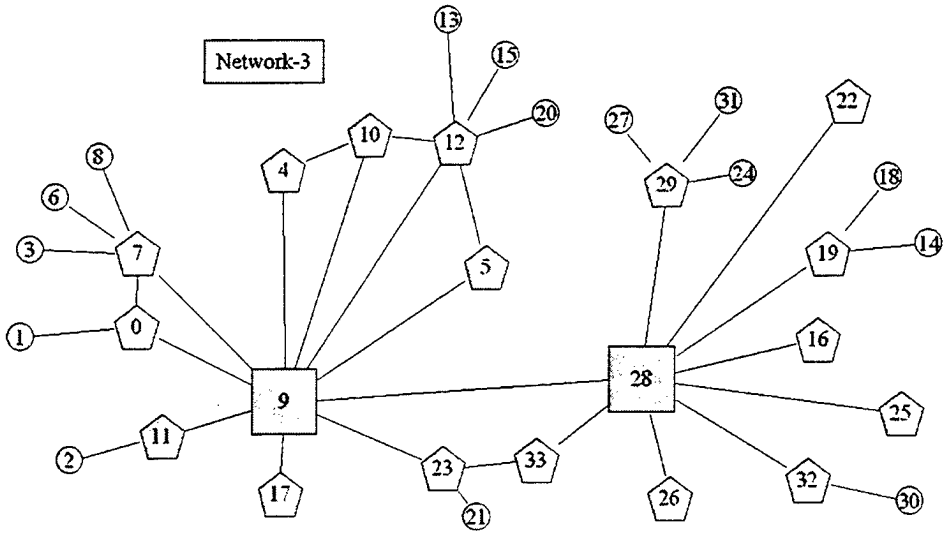
APPENDIX A. TEST NETWORKS

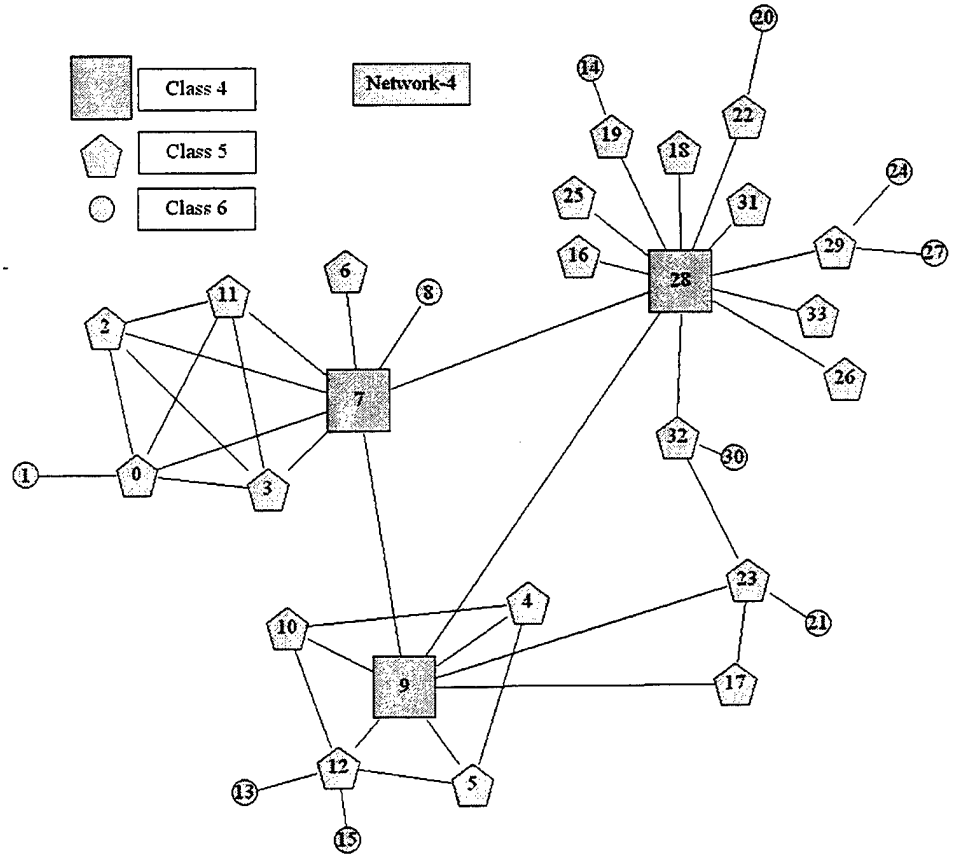
This appendix contains diagrams of each of the test networks, from Olson (1998).

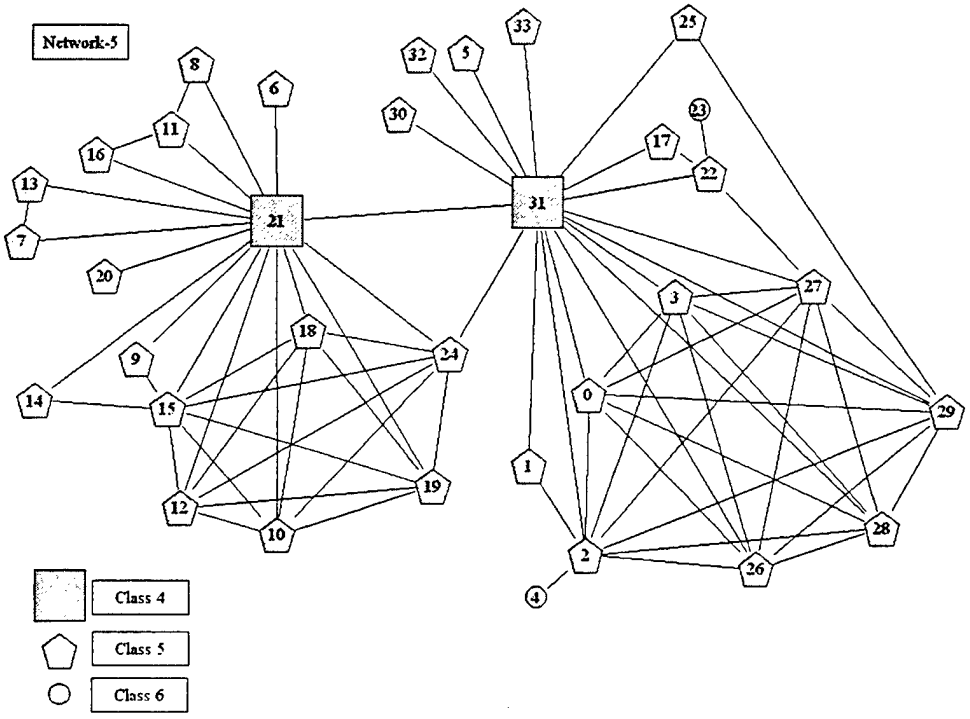


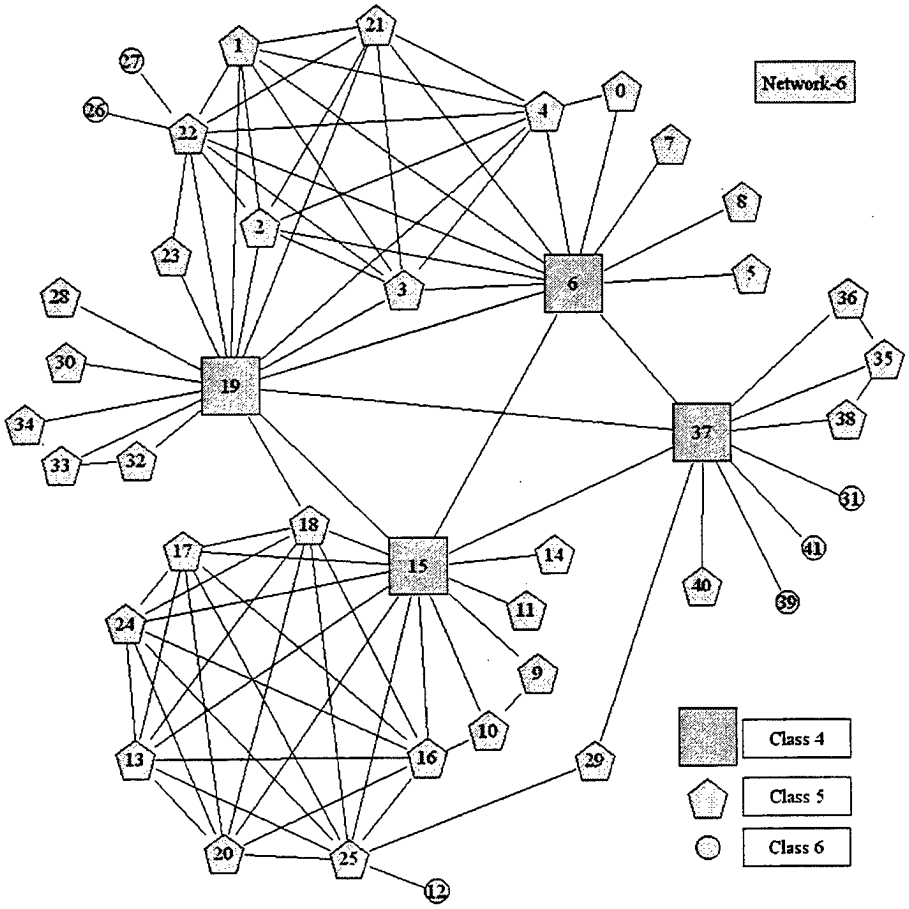


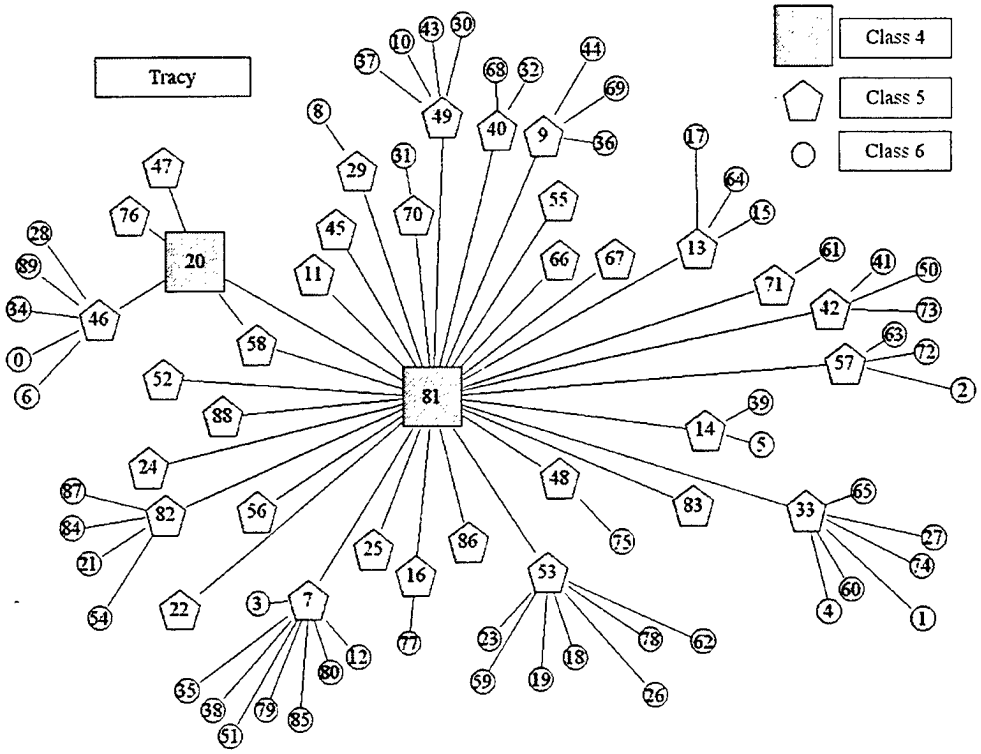

Network-2
Class 4
Class 5
Class 6
14

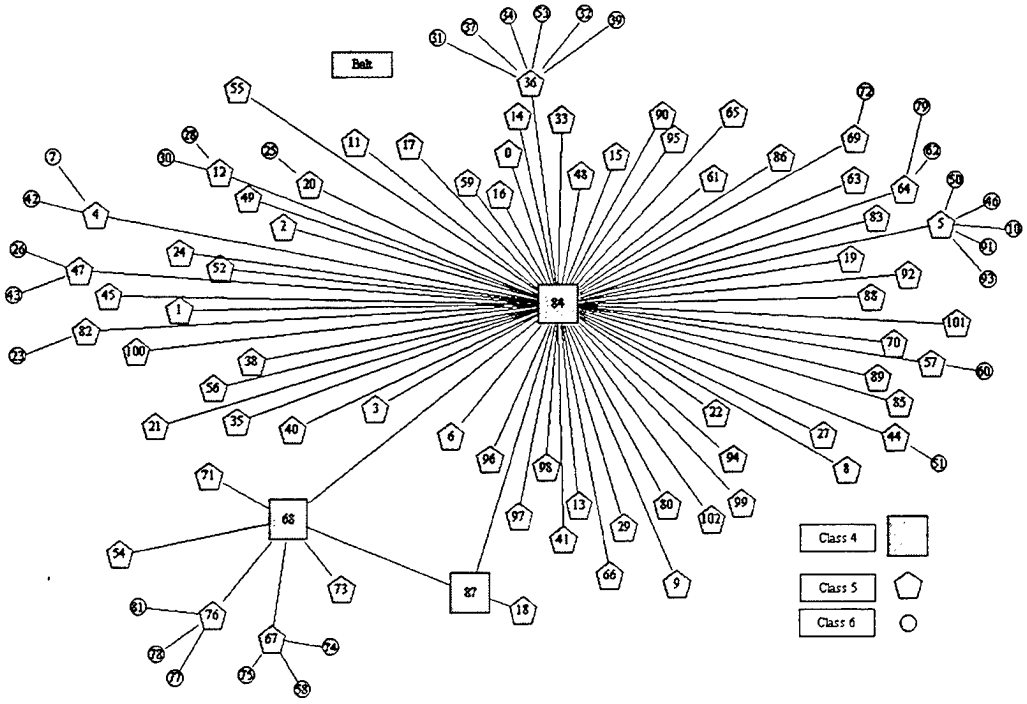

Network-3
9
28


Class 4
Class 5
Class 6
Network-4
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33


Network-5
8
6
11
16
13
7
21
20
18
9
24
14
15
19
12
10
33
32
5
25
30
23
17
22
31
3
27
0
29
1
28
2
26
4
Class 4
Class 5
Class 6


Network-6
Class 4
Class 5
Class 6


Tracy
Class 4
Class 5
Class 6


Belt
Class 4
Class 5
Class 6

APPENDIX B. SOFT INFERENCES

This appendix lists the soft inferences generated for each of the test networks. Net-0 has no soft inferences. P(top) is the probability that a node is a transit exchange. If p(top) > 0.5, the node is most likely a transit exchange. If p(top) < 0.5, the node is most likely a local exchange.

Network-1 Soft Inferences	
node	**p(top)**
0	0.48
1	0.48
2	0.43
5	0.57
6	0.48
7	0.48
9	0.48
14	0.89
16	0.48
17	0.43
18	0.43
19	0.48
20	0.48
22	0.48
26	0.48

Network-2 Soft Inferences	
node	**p(top)**
0	0.48
1	0.48
2	0.43
5	0.57
6	0.48
7	0.48
9	0.48
14	0.89
16	0.48
17	0.43
18	0.43
19	0.48
20	0.48
22	0.48
26	0.48
27	0.55
30	0.43
31	0.55
33	0.57
34	0.57

Network-3 Soft Inferences	
node	**p(top)**
0	0.52
1	0.48
2	0.43
3	0.45
4	0.45
5	0.45
7	0.52
8	0.43
9	0.84
10	0.47
11	0.52
12	0.45
13	0.43
14	0.43
15	0.52
16	0.45
17	0.45
18	0.45
19	0.47
20	0.43
21	0.43
22	0.45
23	0.45
24	0.43
25	0.45
27	0.43
28	0.89
29	0.45
31	0.45
32	0.57
33	0.45

Network-4 Soft Inferences	
node	p(top)
0	0.52
1	0.48
3	0.45
4	0.45
5	0.45
7	0.52
8	0.43
9	0.85
10	0.47
11	0.52
13	0.43
14	0.43
15	0.52
19	0.47
20	0.43
21	0.43
22	0.45
23	0.45
24	0.43
25	0.45
27	0.43
28	0.89
29	0.45
31	0.45
32	0.57
33	0.45

Network-5 Soft Inferences	
node	p(top)
0	0.95
9	0.45
13	0.60
15	0.45
25	0.95
31	0.95

Network-6 Soft Inferences	
node	p(top)
0	0.89
1	0.52
3	0.52
4	0.52
6	0.89
12	0.43
19	0.67
21	0.57
22	0.52
23	0.52
31	0.43
33	0.45
37	0.60
38	0.45
39	0.43
40	0.43
41	0.43

Balt Soft Inferences	
node	p(top)
2	0.60
7	0.60
14	0.60
16	0.45
18	0.75
23	0.60
27	0.45
48	0.45
49	0.75
52	0.55
53	0.60
54	0.60
68	0.80
84	0.80
86	0.45
87	0.75
89	0.45
92	0.45
102	0.77

Tracy Soft Inferences	
node	p(top)
0	0.45
2	0.45
6	0.45
8	0.45
11	0.45
12	0.60
19	0.60
20	0.77
28	0.45
29	0.45
32	0.45
34	0.45
40	0.45
43	0.60
46	0.45
47	0.45
52	0.45
55	0.45
57	0.45
58	0.77
61	0.45
63	0.55
67	0.55
68	0.45
71	0.45
72	0.45
76	0.55
79	0.45
80	0.77
81	0.75
84	0.60
87	0.60
88	0.45
89	0.45

LIST OF REFERENCES

Ahuja, R., T. Magnanti, and J. Orlin, *Network Flows – Theory, Algorithms and Applications*, Prentice Hall, Inc. 1993.

Ash, G. R., *Dynamic Routing in Telecommunications Networks*, McGraw Hill, 1998.

Curet, N., Personal communication, December 1997.

Curet, N., J. Brandeau, and A. Olson, *An Integer Programming Formulation for Estimating Hierarchical Levels in an Undirected Graph.* Presented at INFORMS Conference, Montreal, 29 April 1998.

Executive Order 12333, *U.S. Intelligence Activities*, 4 December 1981.

GAMS Development Corporation, *GAMS Language Guide, release 2.25*, Washington, DC, 1997.

Giarratano, J., and G. Riley, *CLIPS Reference Manual, Version 6.0*, NASA Document JSC-25012, Houston, Texas, 1993.

IBM Corporation, New Orchard Road, Armonk, NY 10504.
http://www.research.ibm.com/osl/osl, accessed September 1998.

Metrowerks, Inc., 9801 Metric Blvd, Suite 100, Austin, Texas 78758.
http://www.metrowerks.com, accessed September 1998.

Olson, A., *Classifying PSTN Switching Stations: A National Security Agency Application,* M.S. Thesis in Operations Research, Naval Postgraduate School, September 1998.

Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303.
http://www.java.sun.com, accessed September 1998.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Professor Robert F. Dell, Code OR/De 4
   Department of Operations Research
   Naval Postgraduate School
   Monterey, CA 93943

4. Professor Gerald G. Brown, Code OR/Bw 1
   Department of Operations Research
   Naval Postgraduate School
   Monterey, CA 93943

5. Norman D. Curet 1
   1452 Graham Farm Circle
   Severn, MD 21144-1085

6. National Security Agency 2
   Center for Operations Research
   Suite 6678
   9800 Savage Rd
   Fort Meade, MD 20755-6678

7. CDR John F. Brandeau 2
   USSTRATCOM, J331
   901 SAC Blvd
   Offutt AFB, NE 68113-6020