



FINAL SUMMARY REPORT

Submitted to:

**SPAWAR Systems Center San Diego
53560 Hull Street, Code 2206
San Diego, California 92152-5001**

Attention:

**Mr. Mark Tukeman, Code 2712
(619) 553-1510**

In fulfillment of the requirements for:

**Contract No. N66001-02-D-0039
*Development of Port Related Transportation Technologies
to Advance Military Responsiveness to National Needs***

**Task Order No. 0012
*Port of Entry and Exit Inspection Process
and Technology Infrastructure, Phase III***

Security Classification: Unclassified

Prepared and Submitted by:

**Center for the Commercial Deployment of Transportation Technologies
California State University, Long Beach Foundation
6300 State University Drive, Suite 220 • Long Beach, CA 90815 • 562.985.7394**

Approved for public release: distribution is unlimited

December 15, 2005

Final Report

Inspection Technology

Seaport Inspection Process and Technology Model

Technical Report consists of the following:

- Description of Suspect Container Targeting System
- Description of Demonstration Software.
- Reviews of Demonstration Plan and Demonstration Report
- Documentation of Demonstration

Prime Contract Number: N66001-02-D-0039
Task Order Number: 0012
Program Element No: 1.18

Submitted to:

Center for the Commercial Deployment of Transportation Technologies
California State University, Long Beach Foundation
6300 State University Drive, Suite 332
Long Beach, CA 90815

Stan Wheatley, Principal Investigator
Subcontract #S07-264402

Submitted by:

College of Engineering
California State University, Long Beach
Long Beach, CA 90840

Table of Contents

Executive Summary 1

1 Introduction..... 2

 1.1 Test/Inspection Objectives..... 2

 1.2 Items Tested/Inspected..... 2

 1.3 Test/Inspection Requirements..... 3

 1.4 Summary..... 3

 1.5 Reference Documents 4

2 Threat Scenario and Security Strategy Rationale for SCTS 5

3 Seaport Inspection Process 7

 3.1 EDI 309 Database 8

 3.1.1 Understanding the EDI 309 Database..... 9

 3.1.2 Data exchange between EDI and ASA program..... 13

 3.2 Advance Simulated Annealing Software 20

 3.2.1 General ASA Description 20

 3.2.2 ASA Modifications for Container Targeting 22

 3.2.3 Compilation and Execution Instructions..... 23

 3.3 Automated Inspection Technology 25

 3.3.1 Pulsed Neutron Activation..... 25

 3.3.2 Human Detection by X-ray Pattern Recognition..... 27

 3.3.3 Human Detection by Heartbeat Sensor..... 31

4 Seaport Inspection Process Demonstration..... 33

 4.1 Test Environment..... 33

 4.2 Test Results (altered due to sensitive data)..... 34

 4.3 Problems encountered..... 34

 4.4 Future Testing..... 34

List of Figures

Figure 1: Seaport Inspection Process Demonstration 7
Figure 2: Demonstration Software Model 8
Figure 3: Identifying Valid Fields in the EDI 309 18
Figure 4: InData.txt 19
Figure 5: OutData.txt 19
Figure 6: The Terrain of Local Search 21
Figure 7: Using ASA Algorithm to Model Suspect Containers 22
Figure 8: The Initialization of Previous.dat 23
Figure 9: Simulated Annealing in ASA 24
Figure 10: Human Contraband 29
Figure 11: Catalog Classification of Suspects 29
Figure 12: Contour Classifier 30
Figure 13: Results of Contour Classifier 31
Figure 14: Overall Comparison (removed due to sensitive data content)
Figure 15: 10% Random Sampling (removed due to sensitive data content)
Figure 16: 50% Random Sampling (removed due to sensitive data content)

List of Tables

Table 1: 309 Data.....8
Table 2: Codified Statement Structure.....13

List of Acronyms

ASA	Advanced Simulated Annealing
ATF	Alcohol, Tobacco, and Firearms
AVIAN	Advanced Vehicle Interrogation And Notification
CSI	Container Security Initiative
CCDoTT	Center for the Commercial Deployment of Transportation Technologies
C-4	Chemical name for plastic explosive
DHS	Department of Homeland Security
EDI	Electronic Data Interface
GAO	General Accounting Office
GS	Functional Group Header
INS	Immigration and Naturalization Service
IRS	Internal Revenue Service
MeV	Million electron volts
PFTNA	Pulsed Fast and Thermal Neutron Analysis
RDX	Chemical name for plastic explosive
ST	Transaction Set Header
STP	Software Test Plan
STR	Software Test Report
TNT	Chemical name of explosive
WMD	Weapon of Mass Destruction

Executive Summary

The threat scenario of a weapon of mass destruction (WMD) within a container is described in order to develop the need for a Seaport Inspection Process. The general intent of this process is to analyze available data within the goods flow – the Electronic Data Interface (EDI) manifests – with recently evolved software and unique inspection hardware components. The software evaluating the EDI data is an Advanced Simulated Annealing (ASA) algorithm that offers many improvements over the neural-net based U.S. Customs Automated Targeting System. The proposed inspection hardware components are pulsed neutron activation equipment with supporting acoustic sensor system.

The envisioned Seaport Inspection Process differs from the present U.S. Customs process. Instead of evolving from a conglomeration of technologies and law enforcement procedures (ATF, INS, IRS, etc.); it is designed from first principles as a feedback system with no human intervention necessary to detect WMDs. The ASA algorithm initializes itself by selecting a percentage of a manifest's containers, both randomly and on threat assessment by ASA. Selected containers are then physically inspected by automated hardware components, and identified either as containing a threat or as safe. These results are fed back into the algorithm, further improving (educating) its ability to correctly select suspect containers for inspection. Random selection is also incorporated into the process, thereby assuring ASA's learning algorithm will detect changing trends in terrorist tactics. The Seaport Inspection Process is a coordinated response to major concerns of the Department of Homeland Security (DHS), which requires: (1) minimal U.S. Customs personnel involvement, (2) provision of non-subjective threat evaluation, (3) security from human interference, and (4) additional security layer to the Container Security Initiative (CSI).

To provide support for the envisioned Seaport Inspection Process, a demonstration of a software model using the ASA algorithm as its core was conducted. This model is called the Suspect Container Targeting System (SCTS). To test the SCTS, EDI 309 manifest data was obtained and database processes applied to structure a subset of the data for compatibility with ASA. Certain packages within the EDI shipment were "seeded" within the subset to simulate a terrorist scenario. The ASA algorithm, with no preconceived weighting of the EDI information, assigned a threat level to each container in the shipment. An automated hardware model appended to ASA inspected a user-selected number of containers with the highest package threats, as well as the requisite random containers. The results of the inspection were fed back to ASA to update and train the algorithm. The EDI database and software were continually exercised to demonstrate the learning speed of the algorithm. Effects of user inputs as to the percentage of inspected containers and ratio of randomly inspected containers were also examined in the demonstration. A simple graphical user interface was generated for facilitating the demonstration. Judicious use of the demonstration should produce a plan for an eventual physical demonstration at a major U.S. port, and foreign port implementation to "extend the U.S. borders."

1 Introduction

This test/inspection report was written according to DID DI-NDTI-80809B and applies to the Suspect Container Targeting System (SCTS) version 1.0 released August 15, 2003. This software was already tested according to the more appropriate DID DI-IPSC-81438A, and documented in the Software Test Report (STR) released September 15, 2003. The intent of the present DID is to authenticate the demonstration (inspection) of that software. Since the software is a PC based model and can be transmitted to SPAWAR electronically, authentication will be accomplished by providing SPAWAR with the software for demonstration at the convenience of SPAWAR. A copy of the software model will thus accompany the submission of this report. Report Section 3.2.3 contains detailed instructions on how to install and run the SCTS software.

The portion of DID DI-NDTI-80809B relative to testing, also covered in DID DI-IPSC-81438A, will comprise a summary of the Software Test Report. Background information on the evolution of the Suspect Container Targeting System Software and the plan on how to test it, is included in this report to produce a self-contained document.

To maintain the structure of the present DID, the “Test” portion of the Test/Inspection Report is taken to mean verification of software performance, while the terms “Inspection” and “Authentication” are taken to refer to providing SPAWAR with a copy of the model software.

1.1 Test/Inspection Objectives

The purpose of the Suspect Container Targeting System (SCTS) is to model the envisioned Seaport Inspection Process of cargo at the ports. Given the manifest information provided to Customs officials in the EDI 309 form, SCTS identifies suspect containers from the data parameters. The algorithm has the capability to improve upon the basis of selection by learning from the results modeled. The foundation of this software is the Adaptive Simulated Annealing for Nonlinear Systems (ASA), a C language program that finds the best global fit of the data in a complex environment. The primary objective of this Test/Inspection process is to provide SPAWAR with a working copy of the SCTS software.

1.2 Items Tested/Inspected

The SCTS is the original version of the software specific to the application of suspect container identification. The Adaptive Simulated Annealing (ASA) code is Copyright (c) 1993-2001 Lester Ingber. This open source program was modified to accommodate EDI 309 data and to assign a threat factor (.01 to .99) to each container in the manifest. Addition of a computer model of the centralized inspection facility completed the process model to form a single software entity, the Suspect Container Targeting System (SCTS). The SCTS software was prepared for the Center for the Commercial Deployment of

Inspection Technology: Final Summary Report

Transportation Technologies (CCDoTT) at California State University Long Beach (CSULB) Foundation.

1.3 Test/Inspection Requirements

To ensure a consistent environment during testing, the following factors were held constant:

- i) The input data was taken from a single EDI 309 manifest (see “Definitive Requirements for the Generation of Output to ASA by the EDI 309 Process,” section 3.1.2).
- ii) A single random model of ten containers, used to seed all test cases
- iii) The number of containers used to create an initial model for ASA
- iv) The size of the model used by ASA, set to five parameters

Changes to preset values of these variables can be made in the file *user_cst.cpp*.

According to the Software Test Plan, the number of containers inspected by the modeled equipment (Variable One) was to increase exponentially from two (2) to eight (8), a reasonable upper bound on the number of containers that could be inspected in a real world scenario. The percentage of random containers opened (Variable Two) was to increase monotonically from 10% to 50%, with a 10% increase on each test. Note, fractional percentages were always rounded up; therefore, at least one random package was always opened; thus producing redundancy in a number of the proposed tests. Variable Three (the number of iterations) was to increase monotonically from one (1) to an upper bound of fifty (50). Variable One has three different values while Variable Two has five different values. Thus, a total of $3 \times 5 \times 50 = 750$ iterations were anticipated. Due to the aforementioned redundancies and inefficiency of running extended iterations on some variable combinations, not all iterations were completed, but the necessary final results were achieved and are described in the body of this report.

Each test is dependent on the availability of EDI manifest data (see The Definitive Requirements for the Generation of Output to ASA by the EDI 309 Process) and the initial model of suspect containers. While the data may change through time, each test trial must use the same data. The introduction of new data may lead to additional rounds of inspection until the algorithm converges upon the selection parameters.

1.4 Summary

The results of the tests run with the SCTS are described in the body of this report. The rationale for generating the model, the particular elements of the software composing the model and the testing of the model are also detailed in appropriate sections of this report. The resultant model is presently being used to evaluate the validity of the Seaport Inspection Process described above. Preliminary results indicate that the “learning” capability demonstrated by the SCTS will provide a marked improvement in the

Inspection Technology: Final Summary Report

container inspection process, resulting in increased port security if such a process is implemented.

1.5 Reference Documents

Three (3) documents are referenced herein; (1) the Definitive Requirements for the Generation of Output to ASA by the EDI 309 Process, (2) the ASA repository located at <http://www.ingber.com/#ASA-INFO>, and (3) the Inspection Technology Software Test Plan, Contract # N66001-02-D-0039, Task Order 12, Program Element # 1.18.

This test/inspection report has been prepared to provide software test results and demonstration verification for the Suspect Container Targeting System (SCTS) version 1.0. This document is intended only for distribution authorized to DOD components only. Other requests shall be referred to:

Space and Naval Warfare Systems Center
Code 2027
San Diego, CA 92152-5001

2 Threat Scenario and Security Strategy Rationale for SCTS

As rationale for a Security Inspection Process, and hence the Suspect Container Targeting System, background information is provided in this section of the report.

A report prepared by the National Defense University's Center for Technology and National Security Policy states that an ocean container itself is ideally suited to deliver a Weapon of Mass Destruction (WMD). The likelihood that a terrorist will use a container to deliver a WMD depends on the type of WMD and the likelihood that an ocean container would be used as the means of delivery.

These researchers believe that it is feasible for a terrorist group to make a radiological "dirty bomb" which uses standard explosives to disperse radiological material, and that the ocean container would provide an ideal mode of transportation. On the other hand, these researchers have concluded, a terrorist attack using a nuclear WMD has a much lower feasibility because it is deemed less probable that terrorists have the resources and technical ability to build or obtain a workable nuclear weapon at this time and the nuclear WMD might be too valuable an asset to relinquish control of by shipping in a container. But some experts agree that the possibility of terrorists smuggling a nuclear WMD by ocean containers merits attention because the consequences would be much more severe than those of other types of WMDs. While there have been no known incidents of containers being used to transport WMDs, criminals have exploited containers for other illegal purposes, such as smuggling weapons, people, and illicit substances. Such activities demonstrate the vulnerability of the freight transportation industry and suggest opportunities for further exploitation of containers by criminals, including terrorist groups.

Various experts have estimated that the cost to the U.S. economy of port closures due to the discovery or detonation of WMDs could be significant. For example, in May 2002, the Brookings Institution estimated that costs associated with U.S. port closures resulting from a detonated WMD could amount to \$1 trillion. Estimating the cost of discovering an undetonated WMD at a U.S. seaport, Booz, Allen and Hamilton reported in October 2002 that a 12-day closure would cost approximately \$58 billion.

The United States Customs department initiates security precautions offshore in port cities of countries which ship goods directly to the United States. To prepare a Container Security Initiative (CSI) team for deployment overseas, Customs sends an assessment team to the CSI port to collect information about the port's physical and informational infrastructure and the host country's customs operations. Customs then deploys a CSI team of approximately four to five Customs officials to work with the host country's customs administration to identify high-risk containers departing from these ports for the United States. Containers targeted for CSI inspection arrive at CSI ports by land, rail, or sea en route to the United States. The CSI team uses Customs' *Automated Targeting System* to screen container data and identify high-risk containers for inspection. This system evaluates U.S.-bound cargo manifest data electronically and determines a container's risk

Inspection Technology: Final Summary Report

level. To improve its screening capabilities, the CSI team further analyzes U.S.-bound containers by means of additional data provided by host countries' customs administration. Containers that both U.S. and host customs officials identify as high risk are then inspected manually, although the arrangements do not specify that U.S. Customs officials must be able to observe inspections. A central tenet of the CSI concept is that U.S. Customs inspectors be able to observe and verify the inspections and that all partner Customs administrations accept this tenet.

According to Customs officials, the most important benefits of CSI derive from the co-location of U.S. Customs officials with foreign customs officials. Prior to the implementation of CSI, Customs officials in U.S. ports screened container data using the *Automated Targeting System* and inspected high-risk containers on their arrival in the United States.

With this overview of the present state of the response to 9/11 by the U.S. Customs it is clear that the primary tool on which U.S. Customs relies is the *Automated Targeting System*. Clearly using a software agent to examine the volume of data available in EDI manifest data should be the foundation of any container security system that strives to minimize impact on goods movement. However, some questions as to the effectiveness of the *Automated Targeting System* have been raised relative to the age of the algorithm in light of more sophisticated software having been developed over the last decade. Also, the algorithm was originally designed for detecting drugs and contraband rather than WMDs. Lastly, very little is published as to how the algorithm is upgraded to learn from the most relevant data source – the immediate results of container inspection. A stronger targeting algorithm would provide the foundations on which to build a Seaport Inspection Process – as is described in this document. Because of its importance and the lack of definitive description of the *Automated Targeting System*, the General Accounting Office (GAO) is currently assessing Customs' *Automated Targeting System* and its overall ability to identify and process cargo containers considered to be “high risk” for terrorism.

3 Seaport Inspection Process

The Seaport Inspection Process was defined and consists of three strongly connected components: (1) EDI data, (2) Targeting Algorithm, and (3) Centralized Container Inspection Facility. A block drawing of the physical inspection process is shown in Figure 1.

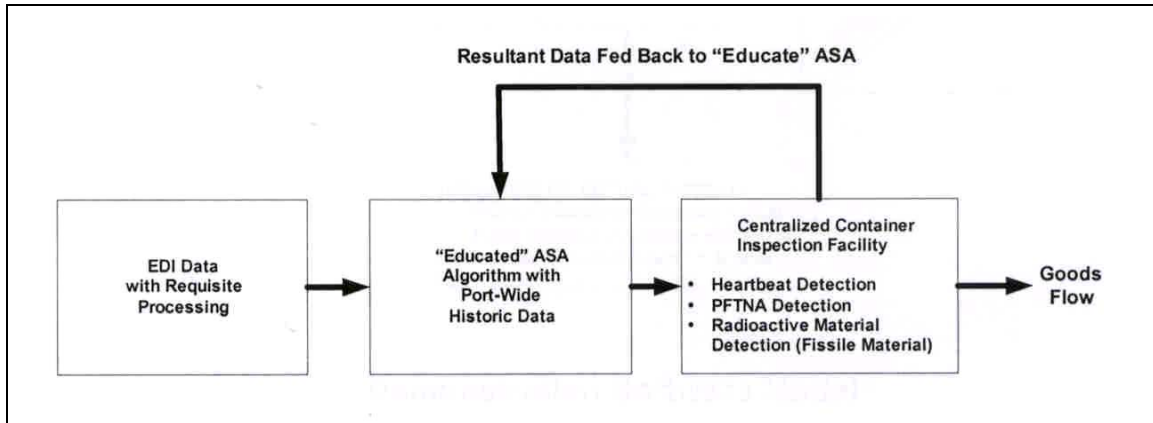


Figure 1: Seaport Inspection Process Demonstration

An actual physical demonstration of such a system is a significant effort involving coordination with U.S. Customs, a terminal or port, and selected inspection equipment manufacturers. Generating a software simulation model of the key elements of the Seaport Inspection Process will provide a means for a security process designer to vary a number of critical parameters to determine the potential effectiveness of the process prior to the expensive physical demonstration process. This software demonstration tool is graphically depicted in Figure 2.

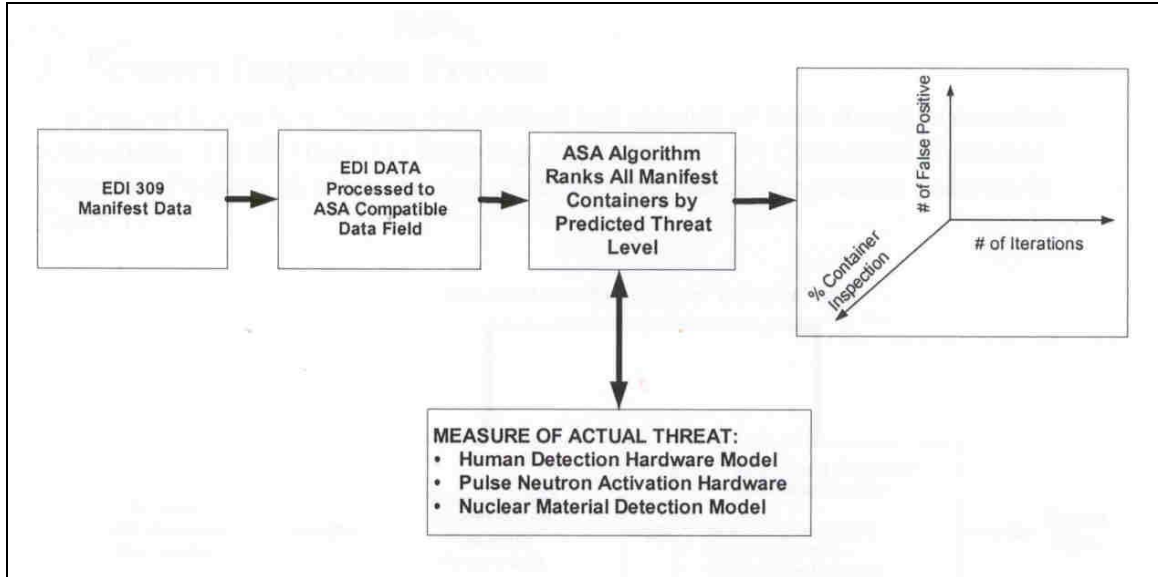


Figure 2: Demonstration Software Model

To generate this software demonstration, historic EDI data was parsed and reduced into a relational database format. Conversion required a number of “catch calls” to allow for abbreviated and missing data fields. The customized open source Advance Simulated Annealing algorithm was tested with a generic data file to verify performance. Simulation of the inspection hardware is focused on Pulsed Fast/Thermal Neutron Analysis (PFTNA) coupled with the Advanced Vehicle Interrogation And Notification (AVIAN) pre-processing system as the primary automated means of determining container risk. This equipment, more fully described in section 3.3, determines the presence of a particular material (explosive, nuclear, biological, etc.). The results of the inspection process are simulated by “seeding” information in certain parcels of the EDI database to model detection of suspect material in that parcel.

3.1 EDI 309 Database

A manifest is a document that lists in detail the total cargo of a vessel and is issued by a carrier or its agent or master for a specific voyage. Examples of data elements in a manifest include shipper, consignee, point and country of origin of goods, export carrier, port of lading, port of discharge, description of packages and goods, and date of lading. The 309 data that was presented to us came in series of cryptic lines, of which a small sampling is shown in Table 1.

Inspection Technology: Final Summary Report

BG*KKLU*AMSKAM*KKLU*USCS*021203*0003*25952...
GS*SO*KKLU*USCS*021203*0004*25952*X*003050...
ST*309*9873874...
M10*KKLU*O*DE*9141273**027E*1*78*W*L*Y...
P4*3001*021210*78...
LX*1...
M11*826614524*57078*6521*PCS*24499*K*122*X*20*CN**KKLU...
N1*CN*WORLD KITCHEN INC...
N3*ONE PYREX PLACE*PO BOX 1555...
N4*ELMIRA**14902...
VID*CN*TRLU*5015485*BZ61768**04000*00000806*00000800*4500*L*HH...
N10*1959*TOYS "SHIPPER ADVISES SHIPMENT CON*MARKS & NUMBERSAS PER ATTACHED SHEE...
N10**TAINS NO SWPM"*T...

Table 1: 309 Data

The only insight for decoding this data available to us was contained in the CCDOTT report, “Information Technology Survey Report”, Project 07 244 301, dated August, 2002. The thrust of this report was to provide a comprehensive view into the EDI milieu. It provided templates that gave insight into the decoding process, but did not provide comprehensive interpretation since that was outside of its scope. As stated in section 9.0, “... (this) print out only shows the first level of data segments. For each data segment additional information is available. For some data segments there are 3 levels.”

Therefore, most of the decoding effort proceeded on an empirical basis.

3.1.1 Understanding the EDI 309 Database

What was disclosed by the report was that each line of data was introduced by a series of characters, up to the first asterisk, which identified the line and by inference, its format. Each line is called a segment, and thus these are known as the segment identifiers.

What had to be discovered was how fields were delimited, how lines were delimited, and how extra-long fields were handled. Even more importantly, it was discovered that these segments had a predictable relationship to one another. This relationship is best described as the Chinese Box model. This model describes the fact that, properly ordered, segments – or groups of segments – are “nested” within other segments. It was also discovered that there are considerably more than 3 levels of segment nesting.

The following are the most significant groups and the relationship between them:

- Header, which begins each distinct transmission.
- Transaction Set. One or more Transaction Sets may be transmitted under cover of one header. In the sample data, there is only one Transaction Set, namely the EDI

Inspection Technology: Final Summary Report

309 Customs Manifest transaction. It is not expected that more than one transaction set under one header will be received.

- Manifest of unloaded goods. One or more manifests may be transmitted within each Transaction Set. In the sample data, there is only one manifest, but there is no reason to expect this will always be the case.
- Port, where the goods are unloaded. One or more ports may be covered by one manifest. In the sample data, there are only goods for Long Beach, and would filter out other ports (i.e. Port of Los Angeles) in the future.
- Bill of Lading. One or more bills of lading may be included for each port under each manifest. In the sample data, there are 78 bills of lading for the port of Long Beach under Manifest number 1.
- Conveyance. Conveyances are such pieces of equipment as trucks, trailers, boxcars, and – for the data – 20-foot or 40-foot Containers. One or more conveyance may be identified in each bill of lading. In the sample data, up to 9 containers are identified by a single bill of lading. A total of 128 containers were identified in the sample data for all bills of lading.
- Content, the detailed description of the contents of a conveyance. These are the deepest nested of the “Chinese box” segments.
- Footer, which concludes each distinct transmission.

With this introduction in place, one can proceed to the next level of understanding of the EDI data, beginning with Transaction Sets.

A transaction set is the collection of data that is exchanged in order to convey meaning between the parties engaged in electronic data interchange. A transaction set is composed of a specific group of segments that represent a common business document (in this case, a Customs Manifest).

Each transaction set consists of the transaction set header (ST) as the first segment and contains at least one data segment before the transaction set trailer (SE). The transaction set identifier is a reference number for the transaction set and is the first data element of the transaction set header segment (ST). This identifier is unique for each transaction set. For Customs Manifests, the identifier is “309”.

Each transaction set is assigned a functional group identifier code. This identifier is the first data element of the functional group header segment (GS). The applicable functional group identifier is also shown at the top of each transaction set after the transaction set name.

As noted above, a transaction set is comprised of a series of data segments. A segment is the intermediate unit of information in a transaction set. Segments consist of logically related data elements in a defined sequence. Segments have a unique segment identifier comprised of a combination of two or three letters or digits that occupies the first character positions of the segment.

Inspection Technology: Final Summary Report

When segments are combined to form a transaction set, their relationship to the transaction set is defined by a segment requirement/condition designator and a segment sequence. Some segments may be repeated, and groups of segments may be repeated as loops.

In the descriptive template as shown in the report, a segment is assigned one of the following three requirement designators defining its need to appear within the transaction set. The requirement designators shown below are each followed by their code abbreviation in parentheses.

- **Mandatory (M)**
This segment shall appear in the transaction set at least once (see Section 2.7).
- **Optional (O)**
The appearance of this segment in the transaction set is either at the option of the sending party or is based on the mutual agreement of the interchange parties.

Each segment is assigned a specific position in the segment sequence of a transaction set, as denoted in the templates by the number to the left of the segment identifier. Within a transmission, segments appear in this order.

Segments with segment requirement designators Mandatory (M) or Optional (O) may be used one or more times at their specific location in the transaction set.

The segments comprising a loop are denoted by a bracket. Within each bracket, on a shaded line, a loop identifier and the number of times the loop may be used at that specific location in the transaction set (“Loop Repeat”) are noted. The Symbol >1 indicates that a loop may be used once or more than once. Loops may be nested within other loops.

If the requirement designator of the first segment in a loop is Mandatory (M), then at least one iteration of the loop is required. If a loop is used, the first segment shall be used for each iteration of the loop. Mandatory segments within a loop are mandatory only if the loop is used.

Next, consider data elements. A data element can represent a qualifier, a value, or text (such as a description). Each data element in a segment is preceded by a separator character, or, in the case of composite data structures, a sub-element separator. The data element separator and sub-element separator must be different from the segment.

If optional data elements occurring at the end of a segment are not used, the data element separators are not transmitted. If optional data elements at the end of a composite data structure are not used, the sub-element separators are not transmitted.

Data elements are assigned a specific position within a segment. In a transmission, the segment is terminated after the last data element used. In this case, the transmission of the segment terminator signifies that the remaining data elements have been omitted.

Inspection Technology: Final Summary Report

The omission of data elements other than at the end of a segment is signified by successive data element separators or, in the case of a composite data structure, sub-element separators.

Each data element in a segment is assigned a number to indicate its sequential position within that segment. The counting of positions starts with 01 for the first data element and is incremented by one to the end of the segment.

Each data element in a composite data structure is identified by a suffix appended to the reference designator for the composite data structure of which it is a member. For example, if the fourth position in the BEG segment were occupied by a composite data structure which contained three data elements, the reference designator for the second data element in the composite would be BEG04-02.

A data element is assigned one of the following three condition designators specifying its need to appear within the segment. The condition designators shown below are each followed by their code abbreviation in parentheses.

- Mandatory (M) This element is required to appear in the segment.
- Relational (X) See Section 2.9.
- Optional (O) The appearance of this data element is at the option of the party or is based on the mutual agreement of the interchange parties.

Relational conditions may exist between two or more data elements within a segment based on the presence or absence of one of those data elements (presence means a data element must not be empty). Relational conditions are specified by a condition code and the identity of the subject elements.

Within a segment, the existence of a relational condition is indicated by the designator “X” in the attributes column. The relational condition is displayed under the heading “Syntax Notes.” The number on the left identifies the reference designator of the first data element of the relational condition. A statement of the conditional relation is provided in two forms: a code and verbiage to explain the code.

The code is constructed using an uppercase letter (P, R, E, C, or L) followed by the two-digit number of the data element reference designator for each data element included in the relational condition. For example, P0102 represents a paired relation between the first and second data elements of the segment. The codified relation statement is constructed as follows. The first character represents one of the following relations.

Inspection Technology: Final Summary Report

CODE	CONDITION	REQUIREMENT
P	Paired or Multiple	If any data element specified is present, then all must be present.
R	Required	At least one of the data elements specified must be present.
E	Exclusion	No more than one of the data elements specified may be present.
C	Conditional	If the first data element specified is present, then all others must be present. However, any or all data elements not specified as the first may appear without requiring that the first data element be present.
L	List Conditional	If the first data element specified is present, then at least one of the others must be present. However, any or all data elements not specified as the first may appear without requiring that the first data element be present.

Table 2: Codified Statement Structure

Comments are provided to clarify or aid in understanding the intended use of the segment.

The EDI conversion process that is now in place incorporates all of these discoveries and appears to be complete in all aspects. Many of the data fields are in fact codes that convey special meaning to the user. The interpretation of these codes required, in many instances, that various government and institutional web sites be queried. Confidence exists that all codes are understood and will be translated properly.

3.1.2 Data exchange between EDI and ASA program

The information SCTS uses to learn patterns of suspect container attributes is reported in an EDI 309 manifest. Two text files are generated to conform to the requirements of the ASA learning algorithm within SCTS; these are:

- InData.txt – reflecting the historical experience with all EDI 309 fields
- OutData.txt – representing the current set of EDI 309 field values.

An EDI 309 may contain omissions, but since the learning algorithm currently does not handle unspecified values not all EDI 309 data is forwarded to SCTS. The concepts of Valid Field and Valid Record are used to filter the data. Valid fields are those for which non-blank data is received for at least 80% of the packages. Valid records are those which contain only valid fields, and more restrictively contain only non-blank valid fields. The application of these concepts will be indicated in the Specification, below.

Specification

- **History**
Internally, the process shall maintain a history of all EDI 309 values ever processed. This history will be used during the preparation of the two text files

Inspection Technology: Final Summary Report

(InData and OutData). Therefore, during the processing of an EDI 309 set of values, the history must be updated prior to the time the text files are generated.

This history shall differentiate between two types of data fields: Continuous, and Discrete.

1. Continuous

Continuous fields are always numeric values. They are fields whose values may fall anywhere between two endpoints, and may in fact be unbounded. The history shall contain one entry for each EDI 309 Continuous field. Each Continuous field entry in the history shall indicate the lowest value ever experienced, including zero, and shall also indicate the highest value ever experienced.

2. Discrete

Discrete fields are alphanumeric, i.e. may be alphabetic, numeric, or a combination of alphabetic, numeric, and special characters. These are fields whose set of acceptable values is usually discontinuous and limited. The history will contain separate entries for each distinctly different Discrete field value that has ever been experienced. For each Discrete EDI 309 field, its distinct values will be sequentially numbered in the order they are experienced, beginning with zero. That is, as a new distinct value for a given field is added to the history, it will be assigned the next higher sequence number for that given field. These sequence numbers are referred to as the field's Index.

- **Indata.txt**

Indata directly reflects the content of the data History, described above.

1. Content

Indata reports data in two groups, in this order:

- a. A single record containing the number (i.e. count) of EDI 309 data fields contained in both InData records, and also contained in Outdata records,
- b. Multiple history data records, one per data element, reported in the following order:
 - (1) the numeric manifest number. (It should be noted that the Manifest Number in each EDI transmission, and hence process, is likely to be a "1"),
 - (2) the numeric container number,
 - (3) all other data fields. It should be noted that since the History contains at least one value for each EDI 309 field, that "History fields" and "EDI 309 fields" are synonymous,

2. Format

Indata will be comprised of a header line, plus one data line for each History field, each terminated with the standard CRLF (CarriageReturn, LineFeed) character pair.

Inspection Technology: Final Summary Report

- a. the header line contains a single data element, the number of fields reported in the balance of file. Since each data element occupies a separate line in the file, this equals the total number of lines in this file less one (the header line).
- b. each subsequent data line, terminated with a CRLF, contains information about each history field, reported in segment pairs:
 - (1) Data Type Identifier.

This is a single letter, followed by a single space. This value is the letter “c” if the field is Continuous, and the letter “d” if the field is discrete.
 - (2) The data-value surrogate.
 - (a) For a continuous field, the surrogate is the highest value experienced relative to the lowest value experienced; that is, it is the numeric difference between the field’s high value and its low value.
 - (b) For a discrete field, the surrogate is the count of the number of distinct entries the field has in the History.

3. Order

A specific ordering of the fields reported in InData is not a requirement, except that fields in InData and in OutData must be reported in the same order. Although not required, all Continuous fields will be reported before the Discrete ones.

4. Filter

The number of fields that appear in InData will be limited to those that pass the Valid Field test. Valid fields are those for which non-blank data is received (during the current processing cycle) for at least 80% of the packages identified in the manifest. This correctly implies that an empty (non-reported, blank) data field is Valid, so long as it is only empty for less than 20% of the packages in the manifest.

- **Outdata.txt**

Outdata directly reflects the current set of EDI 309 values.

1. Content

Outdata reports all data elements as they pertain to each EDI 309 package, one package per output line, subject to Valid Field and Valid Record filtering. The data content of each line is reported in four parts, in this order:

- a. the numeric manifest number. (It should be noted that the Manifest Number in each EDI transmission, and hence process, is likely to be a “1”),
- b. the numeric container number,
- c. all other Valid data fields,

Inspection Technology: Final Summary Report

- d. two random numbers, each between 0 and 1, for internal use by ASA. All random numbers on all output lines are each separately generated random numbers.

2. Format

a. data fields

Each filtered data field in the package is reported as a data-value surrogate, followed by a single space.

(1) for a continuous field, the surrogate is the actual numeric data field value

(2) for a discrete field, the surrogate is the index where the data value is found in the History. The index is the relative number, within all history values for the same data field, associated with the discrete data value being reported. Indexes are always numbered beginning with zero for each data field.

- b. The two random numbers for ASA use will be placed, separated by a single space, at the end of the data line following the last data-value surrogate. It will be in the form "0.nnnnnn" or "0.nnnnnnnn".

3. Order

Outdata surrogates are reported in precisely the same field-order as are indata surrogates.

4. Filter

The number of fields that appear in OutData, just as in InData, will be limited to those that pass the Valid Field test. Valid fields are those for which non-blank data is received (during the current processing cycle) for at least 80% of the packages identified in the manifest.

Additionally, each package in OutData is subject to a Valid Record test. This test requires that no data line is to be generated for any package in which any of its Valid data fields is empty/blank. Thus, only packages for which each and every field contains non-blank data will be reported.

These non-blank fields will, of course, be represented by their surrogates as described above.

- **Process**

The processing of EDI 309 data proceeds through the following 8 steps culminating in the actual creation of the InData and OutData text files:

1. Prepare Segment Tables.

EDI organizes its data in separate segments, according to topic. In this step, tables for each of the then current segments (presently 27), must be created and prepared for receiving the EDI data.

Inspection Technology: Final Summary Report

2. Process 309 Raw Data.

The EDI 309 data is deciphered and organized into formal relational data elements and placed into the proper relational tables. Proper foreign-key references are created and placed with each data record so that a properly normalized relational database results.
3. Build Current Distinct Values

Each of the segments is examined and duplicates are removed. The result of this process is a coherent set of distinct values and a count, for each, of the number of times it was repeated.
4. Prepare and Post the History

The set of current distinct values is compared to the History. All current *discrete* values not present in the History are added to it. All current continuous values not present in the History are added to it. In addition, the continuous value entries in the History are updated whenever the lowest of the current-value minimums is lower than the History record for the same field, and are also update whenever the highest of the current-value maximums is higher than the History record for the same field. The result is a single history record, for each continuous field, that contains both the lowest value ever experienced, and the highest value ever experienced.
5. Convert Normalized Tables into a Flat File

A flat file, the equivalent of a huge spread sheet, is created in which there is a column for each of the data fields in each of the approximately 27 segment tables. The columns are created by examining each of the segment tables, identifying each of their columns, and creating counterpart columns in the flat file. This process must always be performed anew, lest the addition of a new field to one of the segment tables be overlooked.

After creating the structure of the flat file, its rows are filled by processing each package: the data elements for the package, its sibling segments, its parent segment, and all its ancestral segments are extracted and placed into the flat file record for that package.
6. Identify Valid Fields

Each column in the flat file is examined to determine which columns (i.e. field) are valid ones, i.e. have an 80% or greater presence in the flat file. Both qualifying and non qualifying columns are identified.
7. Generate InData.txt

Data is extracted from the History and written to the InData.txt file. Only valid data fields are processed

Inspection Technology: Final Summary Report

8. Generate OutData.txt

Data is extracted from the Flat File and written to the OutData.txt file. Only valid data fields are considered, and each record must be determined to be valid before it can be written. Continuous fields are written directly, whereas for each discrete field in each record its relative index must be calculated.

The following is an example of the transformation from EDI 309 data to InData.txt and OutData.txt. In this fictitious manifest: there are three containers, with 5, 8, and 1 package(s), respectively.

MANIFEST ID	CONTAINER ID	FIELD 3	FIELD 4	FIELD 5	FIELD 6	FIELD 7
1	1	234.5	10	.234	0	
1	1	420.0	5	.12	1	23.5
1	1	417.45	13	.789	0	
1	1		19	.234	2	26.7
1	1	345.2	13		0	23.7
1	2	123.5	5		1	
1	2	146.452	9		0	
1	2	119.0	15		1	45.3
1	2	123.4	19		0	45.2
1	2		5	.345	1	
1	2	154.99		.123	2	
1	2	167.23	20	.234	1	
1	2	278.4	14		1	123.9
1	3	387.0	13	.234	1	23.8

Figure 3 Identifying Valid Fields in the EDI 309

Fields 5 and 7 are invalid since they are not specified in >80% of the packages. The packages highlighted in red with the bold italic type are not valid packages. From this information InData.txt can be created. The format of InData.txt is provided in Figure 4 the first line will always provide the total number of usable fields found in any container manifest, *n*. This should be followed by exactly *n* additional lines of code, one for each valid field. As stated earlier, each field is categorized as a continuous or discrete variable. In the case of continuous variables (fields 1, 2, and 3) a marker ('c') and the adjusted upper bound is provided. With discrete fields (fields 4 and 5), the marker ('d') is followed by the total number of categories associated with that field. With the exception of the marker characters 'c' and 'd', the data is always numeric. For this reason white space delineation is used to distinguish fields.

Inspection Technology: Final Summary Report

```
5
c 0
c 2
c 301
d 15
d 3
```

Figure 4: InData.txt

There are only five valid fields in the manifest since fields 5 and 7 were eliminated due to lack of proper reporting. Reminder, the first two fields reported are the manifest and container ids. For all other fields the order does not matter so long as it is consistent between *InData.txt* and *outData.txt*. The upper bounds and the number of categories have been adjusted. For example, since there is only one manifest, the upper bound becomes 0, <1-1>. The upperbound on Field3 is 301.0 <420.0 – 119.0>.

While the information in *InData.txt* provides a historical view of container attributes, the values in the second file are concerned with individual containers. As each container enters the port its EDI data will be sent directly to a file named *OutData.txt* (Figure 5). The purpose of *OutData.txt* is to allow SCTS to compare the characteristics of each package/container to a model of previous packages/containers.

OutData.txt						
0	0	115.5	5	0	.99	.99
0	0	301.0	0	1	.01	.01
0	0	298.45	8	0	.01	.01
0	0	226.2	8	0	.01	.01
0	1	4.5	0	1	.01	.01
0	1	27.452	4	0	.01	.01
0	1	0.0	10	1	.01	.01
0	1	4.4	14	0	.99	.99
0	1	48.23	15	1	.01	.01
1	0	159.4	9	1	.01	.01
0	2	268.0	8	1	.01	.01

“Dirty” container identifier

Figure 5: OutData.txt

Even though there were fourteen packages in the manifest, only eleven (11) are reported since three do not have values for all of the valid fields. Each line specifies the (adjusted) values associated with a single package. The values have been changed to match the category number or adjusted value for a continuous field. For instance, in the first package the value reported in field 3 is 115.5, <234.5 – 119.0>. Note, even though there

are only five valid fields in the manifest, there are seven data points. The last two data items are randomly generated numbers that identify “clean” (value of .01) or “dirty” (value of .99) packages. These final two fields designate the probability that the container holds a dangerous substance; i.e. the results of the pulse neutron activation. This field is not used by SCTS to determine which containers fit the model of suspect containers; it is only used for testing the correctness of the algorithms choices. These numbers may be changed later to fit a specific model specified by the user.

3.2 Advance Simulated Annealing Software

The problem of determining which containers should be checked when ships enter security at a dock is complex. Two methods are possible. One is to check every container on board the ship, the other is to randomly select containers for inspection with the hope that illegal or dangerous material may be discovered thereby. Of course in many cases neither of these approaches is optimal. In a resource-constrained environment, checking every possible container is not feasible. And while randomness has been shown to often be as effective as heuristics in some domains, it is generally agreed that algorithms that incorporate a more informed approach will find a better rate of success.

In the current approach, the beneficial property of randomness is retained; also a learning algorithm is incorporated to help eliminate waste of resources by focusing the heuristic on identifying those containers that exhibit traits that can be analyzed particularly well by the pulse neutron activation detector. This technique is based on a search technique called Adaptive Simulated Annealing, in particular the publicly available ASA program. The search and simulated annealing algorithm is briefly described before the details of the ASA program and the implementation.

3.2.1 General ASA Description

ASA Adaptive Simulated Annealing for Nonlinear Systems (ASA) [Ingber 93] is a global optimization algorithm – based on the physics of a cooling system of particles – that was developed to statistically find the best global fit of a nonlinear constrained cost-function over a N-dimensional space. Instead of particles cooling to form the most efficient structure for a solid, data records are organized to form the most efficient information represented by that data. The algorithm permits an annealing –or cooling -- schedule for a “temperature”, T , decreasing exponentially. For data the “temperature” is analogous to the number of iterations allowed to find relationships between the data. By introducing re-annealing, ASA addresses the need to deal with the changing sensitivities between the data records that occur in a multi-dimensional space. Re-annealing is done by periodically rescaling the annealing time. When the rescaling is performed, the initial acceptance temperature is reset to the maximum of the most current minimum [Ingber 95].

ASA has over 100 OPTIONS to provide robust tuning over many classes of nonlinear stochastic systems. The C-language code is widely distributed and has been used in a

Inspection Technology: Final Summary Report

number of domains, e.g. combat analysis [Ingber, Fuji, Wehner, 1991; Ingber, Sworder, 1991], finance, and large-scale systems [Ingber, 1992]

Direct comparisons have been made between the performance of ASA to Boltzmann annealing (BA), fast annealing (FA), and publicly available genetic algorithms (GA). When compared to BA and FA on difficult test problems provided with the ASA code, only ASA regularly attained the global minimum, and it was more efficient in attaining regular minima than either of the other two algorithms [Rosen 1992]. In the comparison against GA, ASA outperformed the GA in every case of the GA test suite [Schraudolph, Grefenstette 1991].

Consider the case where each container is a node in a graph. The fully enumerated search would require a check of every node. However, in local search, the algorithm finds an initial (possibly low quality) solution quickly and then checks neighboring nodes to determine if they are of higher quality. The search completes when time runs out, or the algorithm has found the highest quality nodes. Hill-climbing is one type of local search, where the algorithm is compared to climbing to the top of a hill by always stepping up, until you have reached the maximum height. A potential downfall of local search is that the algorithm may get trapped on local maxima – areas where all the neighbors of the current solution are of lower quality, but higher quality solutions still exist (See Figure 6).

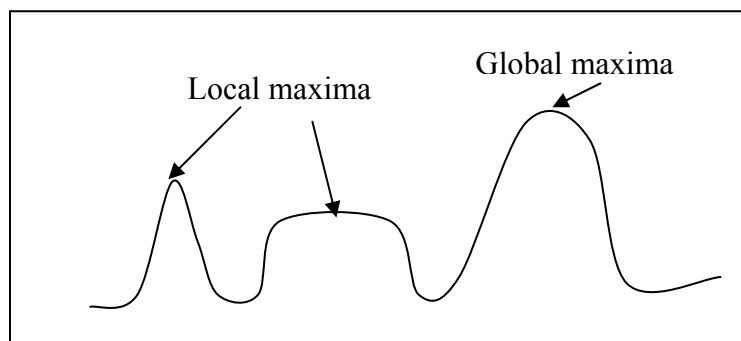


Figure 6: The Terrain of Local Search

Despite the problems with local search, it has proven to be the most effective method for solving many search problems thanks in part to a number of heuristics. One method for reducing the possibility of becoming trapped on local maxima (or as more commonly referenced local minima) is to use simulated annealing. With simulated annealing the algorithm will always climb to a higher neighbor, however if a higher quality solution does not exist, the algorithm will occasionally choose a lower quality node to explore next. The intended reward for this decrease is that the algorithm may next climb to a new hill with a higher maximum. As time progresses, the probability that the algorithm will choose a lower quality node decreases.

3.2.2. ASA Modifications for Container Targeting

As stated earlier, the EDI 309 data and information about previous data is used to train the ASA learning algorithm and develop a model of suspect containers. The first time that SCTS is executed, no previous data exists, so the program randomly chooses a user-designated number of containers for opening. While not implemented in the current system, it would also be possible to seed the initial model with information provided by Customs agents or other users. The results of inspecting these containers will be used to create the very first model of suspect containers. Again, it is important to note that the containers themselves are not actually opened in the virtual environment, but the hidden probability field from the file OutData.txt is used to determine their content.

Once *previous.dat* is created, future compilations of the program use this information to create a new model of suspect containers. This model is created by first choosing parameters and/or combinations of parameters of interest, and then calculating the coefficients for each variable.

Top-level ASA: picks the x parameters (individual and cross-form) to look at for the equation:

$$\ln\left(\frac{y}{1-y}\right) = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 \dots \beta_n X_n$$

Low-level ASA: determines the optimal coefficients to use for β given set of x parameters.

Figure 7: Using ASA Algorithm to Model Suspect Containers

Once the model is created, containers in the current manifest are ranked in descending order of their closeness to the model of suspect containers. Using this ranking the model of the centralized container inspection facility will “investigate” (read the data regarding) the contents of each selected “container” (data packet). The information about the inspected “containers” is then used to update the selection criteria, is time-stamped, and incorporated in the data file. As more information comes in, the older data is replaced by more recent arrivals. The flow of control between the software components and the ASA algorithm is shown in Figure 7.

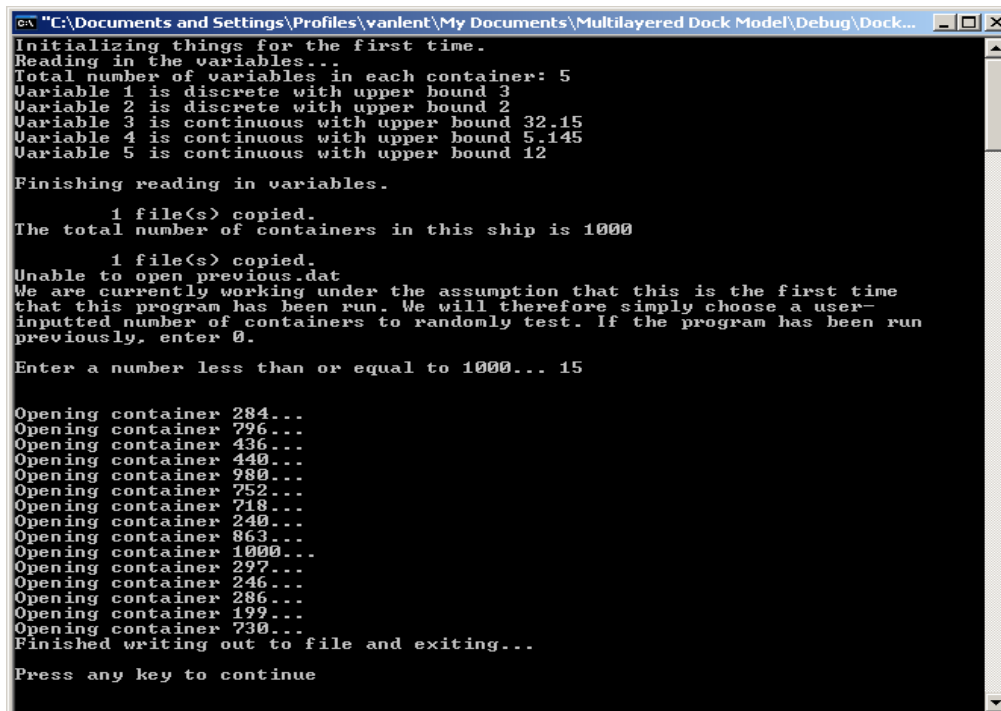
Once the model of suspect containers has been developed, the model can be updated as new data become available. The learning algorithm benefits from two cases, false

Inspection Technology: Final Summary Report

positives and false negatives. A false positive occur when the program flags a data packet representing a container as suspect but upon further inspection is found to be clean. For every false positive, the internal model is updated to reflect this information. False negatives are harder to handle. In this case the algorithm deems a record as “clean” when it really should have been flagged for “inspection” (sent through the automated inspection hardware model). Because a record actually containing data indicating a risk is not flagged, the model is not updated. To catch every false negative, every record would have to be flagged for for inspection (equivalent in the physical world to inspecting every container.) However, because randomness is incorporated into the selection criteria, the occasional “dirty” packet will be flagged and inspected. The model will then be updated to reflect the new parameters.

3.2.3. Compilation and Execution Instructions

The files required for program execution are provided in the Software Test Report (STR). The program is written in C/C++ and has been tested on Windows NT with Microsoft Visual C++ 6.0. The program workspace is *DockModel.dsw*. The file *previous.dat* is included with the code and provides a minimal history. If this file is deleted ASA cannot create a model once the variable formats have been read from *InData.txt*. The user is informed that there is no previous information and is queried for the number of random containers to be chosen for creating an initial model. The manifest information and a timestamp for each randomly selected container will be stored in a newly created *previous.dat* (Figure 8). Obviously, the larger the number of random containers selected, the easier it will be for ASA to create an accurate model.



```

C:\Documents and Settings\Profiles\vanlent\My Documents\Multilayered Dock Model\Debug\Dock...
Initializing things for the first time.
Reading in the variables...
Total number of variables in each container: 5
Variable 1 is discrete with upper bound 3
Variable 2 is discrete with upper bound 2
Variable 3 is continuous with upper bound 32.15
Variable 4 is continuous with upper bound 5.145
Variable 5 is continuous with upper bound 12
Finishing reading in variables.
1 file(s) copied.
The total number of containers in this ship is 1000
1 file(s) copied.
Unable to open previous.dat
We are currently working under the assumption that this is the first time
that this program has been run. We will therefore simply choose a user-
inputted number of containers to randomly test. If the program has been run
previously, enter 0.
Enter a number less than or equal to 1000... 15

Opening container 284...
Opening container 796...
Opening container 436...
Opening container 440...
Opening container 980...
Opening container 752...
Opening container 718...
Opening container 240...
Opening container 863...
Opening container 1000...
Opening container 297...
Opening container 246...
Opening container 286...
Opening container 199...
Opening container 730...
Finished writing out to file and exiting...
Press any key to continue

```

Figure 8: The Initialization of Previous.dat

Inspection Technology: Final Summary Report

Assuming previous data, the normal execution of the program is as follows:

1. Check file “*indata.txt*”
Indata.txt is a text file that lists the number of variables, followed by a description of the variable (categorical or discrete), and the number of categories or the upper and lower bounds of the discrete values.
2. Use *previous.dat* to create model of suspect containers. The parameters, coefficients, and value of the linear equation are displayed. The final value gets progressively lower, but occasional higher values are encountered as the algorithm attempts to avoid local minima. (Figure 9) shows the search performed by ASA. Given 50 containers in the *previous.dat* file, ASA took 174 iterations to converge. The maximum value was 15.4864 and the final value was .000160593.

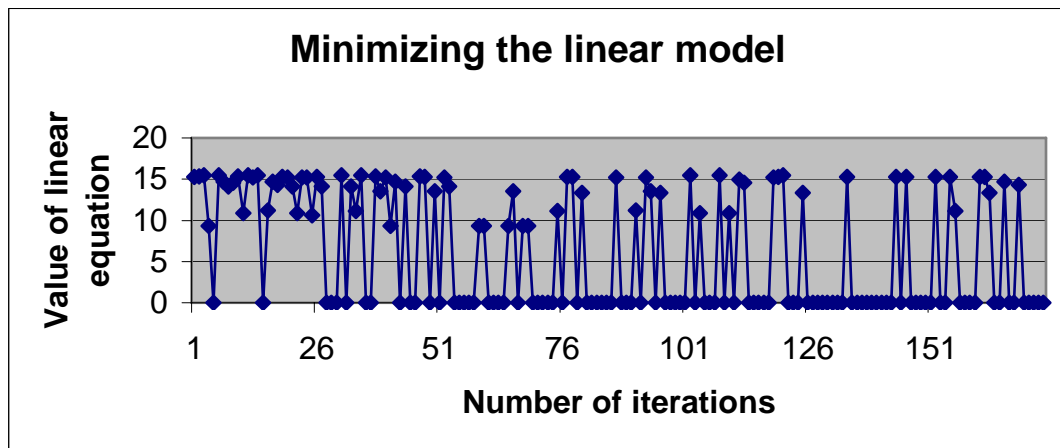


Figure 9: Simulated Annealing in ASA

3. Finally, read in current manifest and sort the containers based on the ASA model. In this case the top six records were “opened;” i.e., checked against the hidden probability. The information on these six containers was then time stamped and returned to *previous.dat*. To see which records were “opened,” check the file for the packets with the lowest timestamp.

Three files are created to provide data recording for analysis:

1. *Results.txt* (previously called FalsePositives.txt)— the average “quality” of container selections based on the results of inspecting the containers. High-quality containers are those that are most likely to contain hazardous material.
2. *History.txt* – a cumulative history of all containers opened since the initial model was created.

Inspection Technology: Final Summary Report

3. *Previous.dat* – a description of the current containers used to update the ASA model.

These values may be imported into statistical software and plotted linearly against the number of iterations to display the learning curve.

3.3. Automated Inspection Technology

Looking for WMDs in a container has created large scale applications of small scale analysis techniques initially intended only for the laboratory. Historically, no means existed to determine *material* content of a container. Visual inspection was the only means to determine a container's contents – not of what it was made. The visual inspection evolved into a non-invasive x-ray inspection which still required a “trained eye” to evaluate the image of container contents, but still no material information. Until recently, the problem of determining a container's material contents required conventional, “hands-on” chemical analysis of the material discovered within the container. So the issue is one of performing an automated chemical analysis on material in the container without opening it.

Fortunately, the analysis techniques recently applied to container contents are automated and require no additional trained individuals to operate.

3.3.1 Pulsed Neutron Activation

High explosives (TNT, RDX, C-4, etc.) are composed primarily of the chemical elements hydrogen, carbon, nitrogen, and oxygen. Many innocuous materials are also primarily composed of these same elements. These elements, however, are found in each material with very different elemental ratios and concentrations. It is thus possible to identify and differentiate, e.g., TNT from paraffin. For narcotics, the C:O ratio is at least a factor of two larger than that of innocuous materials. Explosives have been shown to be differentiated by the utilization of both C:O ratio and N:O ratios. The problem of identifying explosives is thus reduced to the detection of elemental quantities.

Nuclear techniques show a number of advantages for non-destructive elemental characterization. These include the ability to examine bulk quantities with speed, high elemental specificity, and no memory effects from the previously measured object. These qualities are important for an effective detection system for explosives and drugs.

The physical principles that all these methods are based upon have been established for a number of years, and have been extensively used by nuclear physicists and chemists for the investigation of nuclear structure.

In principle, a neutron impinging on an object can initiate one of several nuclear reactions with the chemical elements of which the object is composed. In most of these cases, as a result of these reactions, γ rays are emitted with characteristic and distinct energies. These γ rays are like the “fingerprints” of the elements contained in the object. By counting the number of γ rays emitted with a specific energy (e.g. the γ rays of sulfur),

Inspection Technology: Final Summary Report

one can deduce the amount of the element contained within the object. In the case of an object that is hidden among other innocuous materials, the identification takes place through the correlation of various chemical elements observed, coupled to the information about the innocuous material itself.

Neutrons are highly penetrating particles. Their intensity is not diminished by the thickness of common containers. To a lesser extent, the outgoing γ rays are also very penetrating, easily exiting the interrogated volume to be detected by an appropriate set of detectors placed outside the object. Thus, the method is non-intrusive (the interrogation can take place from a distance of several centimeters) and non-destructive because of the very small amount of radiation absorbed by the interrogated object.

Depending on the chemical elements that one wishes to measure, one might have to use neutrons of several energies. In many of the neutron-based applications currently in use, radio isotopic sources (Am-Be, ^{252}Cf) are utilized for neutron production. These sources can excite a host of chemical elements (H, C, S, Fe, etc.) through neutron capture reactions. However, there are other elements such as C and O which need neutron energies several MeV higher than those available from the radioactive sources. To satisfy this, a neutron source is required that can produce the high energy neutrons for measurement of elements such as C and O, and low energy (0.025 eV) for elements such as H and Cl. It has been shown that such a task can be accomplished with the utilization of a pulsed neutron generator. This technique is called Pulsed Fast/Thermal Neutron Analysis (PFTNA).

The basis of PFTNA is a pulsed neutron generator utilizing the deuterium-tritium (d-T) reaction. The pulsed d-T neutron generator provides 14 MeV neutrons which in turn initiate several types of nuclear reactions ($(n,n'\gamma)$, $(n,p\gamma)$, (n,γ) etc.) on the object under scrutiny. The γ rays from these reactions are detected by a suitable set of detectors (usually bismuth germanate (BGO) scintillators). During the neutron pulse, the γ -ray spectrum is primarily composed of γ rays from the $(n,n'\gamma)$ and $(n,p\gamma)$ reactions on elements such as C and O, and is stored at a particular memory location within the data acquisition system. Between pulses, some of the fast neutrons that are still within the object lose energy by collisions with light elements composing the object. When the neutrons have an energy less than 1 eV, they are captured by such elements as H, N, and Fe through (n,γ) reactions. The γ rays from this set of reactions are detected by the same set of detectors but stored at a different memory address within the data acquisition system. This procedure is repeated with a frequency of approximately 10 kHz. After a predetermined number of pulses, there is a longer pause that allows the detection of γ rays emitted from elements such as Si and P that have been activated. Therefore, by utilizing fast neutron reactions, neutron capture reactions, and activation analysis, a large number of elements contained in an object can be identified in a continuous mode without sampling.

PFTNA uses low resolution, high Z detectors such as bismuth germanate (BGO) or gadolinium ortho-silicate (GSO). Data analysis of the resulting spectra is performed with a spectrum deconvolution code. These detectors can be designed to serve a dual purpose

Inspection Technology: Final Summary Report

of not only looking for the characteristic γ -ray spectra of PFTNA, but also spectra of fissile material

The PFTNA (Pulsed Fast/Thermal Neutron Analysis) technique measures the elemental contents (oxygen, nitrogen, etc.) within small volume segments (voxels) of a scanned object. These measurements can be used to generate three-dimensional maps of the cargo's elemental composition; thus if an automated detection of a threat material occurs, the location within the container can be determined. The amounts and relative concentrations of key elements are used to identify specific substances and set off programmed, automated threat detectors.

Inspected objects (cargo containers or goods-carrying trucks) move through the system and are exposed to short pulses of fast neutrons. When the neutrons hit the cargo container, gamma rays are emitted from the cargo and the cargo container. Gamma detectors located around the inspected containers collect elemental signals emitted from the container contents. An electronics data acquisition system processes the signals and routes the elemental and spatial data to a computer system that generates elemental images of what is present in the cargo container.

These elemental signatures are compared to material signatures which have been programmed into the computer. This comparison enables the system to automatically identify a variety of contraband types. High resolution images display the position and extent of contraband in the cargo container or truck. The entire process is fully automated and does not require human operator interpretation. Detection does not rely on shape and is immune to diligent packaging.

Since the output of the PNFTA is the amount of a particular explosive or chemical compound, the amount of seeded material in the EDI database will be assumed to be proportional to the PNFTA output. Addition of the manufacturer's detection threshold and limited "false positive" information allowed a straightforward relationship between seeded material amount and modeled likelihood of detection by PNFTA.

While human detection is a prerequisite of PNFTA inspection, it, in itself, is also an inspection process for terrorist entry into the U.S.

3.3.2. Human Detection by X-ray Pattern Recognition

In obtaining the software for determining the modeling of human detection in the Seaport Security model, certain assumptions were made that were found not to be valid. First was that the difference in contrast between the human contraband and the background would not be a determining factor in recognition. The second invalid assumption was that enough information was available to properly train the network for all possible derivation of positions and personnel.

Two tools were planned to be used for human detection -- the *Image Classifier* and the *Contour Classifier*.

Inspection Technology: Final Summary Report

The *Image Classifier* tool allows one to classify (identify) images by comparing them to images that were cataloged previously. An example of an application where the *Image Classifier* tool would be useful is the sorting of three different software packages on CDs that are coming down an assembly line. Since the CDs have different labels, the feature that you will try to identify on each of the CDs is the label. All the CDs must be in approximately the same position. For example, you may want to attach the CDs to hubs located in the same position on the assembly line. Note, however, that one can program the tool to take slight changes in position (shifting) into account. This is useful if, for example, the labels of some of the CDs were attached slightly lower or slightly higher on the CD.

In the tool's training stage, one builds a catalog of images, called training images. If the objects one is trying to identify may shift or rotate during the classification stage, one can program the *Image Classifier* tool to automatically create additional training images that take this rotation and/or shifting into account.

To create these additional training images, the *Image Classifier* tool requires a background image as well as a mask image for each of the objects you are trying to identify. These images are described as follows:

- Background image – The background image is an image that represents as closely as possible the background on which the features under test will be acquired. When creating additional training images for objects in a rotated or shifted position, the tool replaces the pixels in the original input image with the pixels from the background image.
- Mask image – The mask image is a grayscale or binary image that represents the object you are trying to identify. You must create your own mask image. The pixels that you are trying to identify must not equal 0. All pixels in the mask image that are equal to 0 (black) are ignored by the tool and are replaced by pixels from the background image.

If one wants to use a binary mask image, it can be created from the input image by using the *Threshold* tool. If one uses a grayscale image, it can be created from the input image by using the *Pixel Change* tool. The additional training images that the *Image Classifier* tool creates are in the same plane as the original input image. If additional training images that represent other planes are required, these training images must be created manually. The more training files the tool must create (based on the number of rotations and/or shifts that you want to allow for) and the more results that you want the tool to report, the longer the tool will take to create the catalog (many hours or several days in some cases) and the more memory the compiled catalog will require.

In the classification stage, a feature under test (an image containing the feature you are trying to identify) is compared to the training images in your catalog. The *Image Classifier* tool returns information, such as the name of the training image in the catalog that best matches the feature under test, the angle of rotation and/or shift in X and Y

Inspection Technology: Final Summary Report

direction of the feature under test with respect to the training image, and a score that measures the confidence level of the match. The classifying of images is performed very quickly.

The *Contour Classifier* tool allows classification of enclosed shapes, called “contours under test”, by comparing them to contours cataloged previously.

To use the *Contour Classifier* tool, one first creates a binary mask image then extracts contours from the binary image, add the extracted contours to a catalog, and name them. Comparison of the contours under test to the contours in the catalog can be made after the catalog is created.

The *Contour Classifier* tool produces the name of the catalog element that best matches each contour under test, three (3) Euler angles (alpha, beta, and gamma) that describe the rotation of each contour under test with respect to a contour in the catalog, and the score, which is a measurement of how good the match is between the contour under test and the contour in the catalog.

A typical example of non-working detection of human contraband is shown in (Figure 10) (Figure 11) is the training picture to recognize features that under test, i.e. the human contraband.

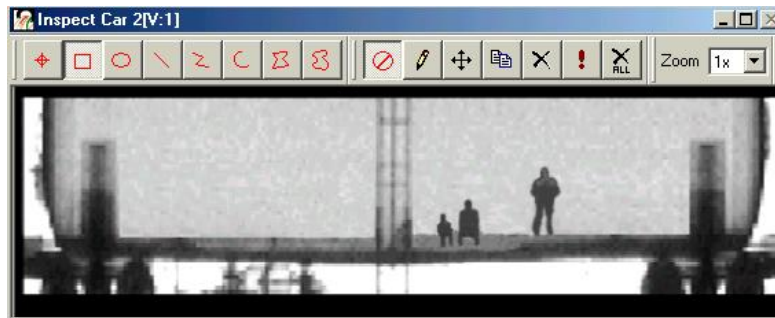


Figure 10: Human Contraband

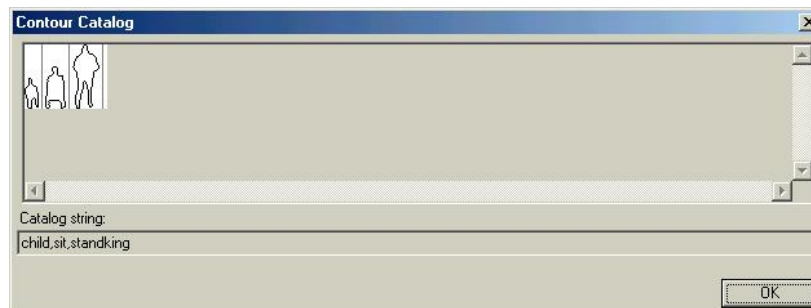


Figure 11: Catalog Classification of Suspects

Inspection Technology: Final Summary Report

Each individual picture must have its threshold manually adjusted to bring out the workable contrast. This is due to the fact that each picture does not have the same resolution or light intensity. Human contraband wearing winter clothing contrasts differently than the same person wearing summer attire. Therefore when the catalog is made this variation must be taken into consideration thereby making it necessary to create more training pictures than originally anticipated. A training picture must be made of each possible variation in both the amount of clothing worn by the suspects and the position of the individual compared to their background. The contour classifier does not recognize objects below a settable level. This makes the detection of a human cargo hidden inside a crate impossible to detect.

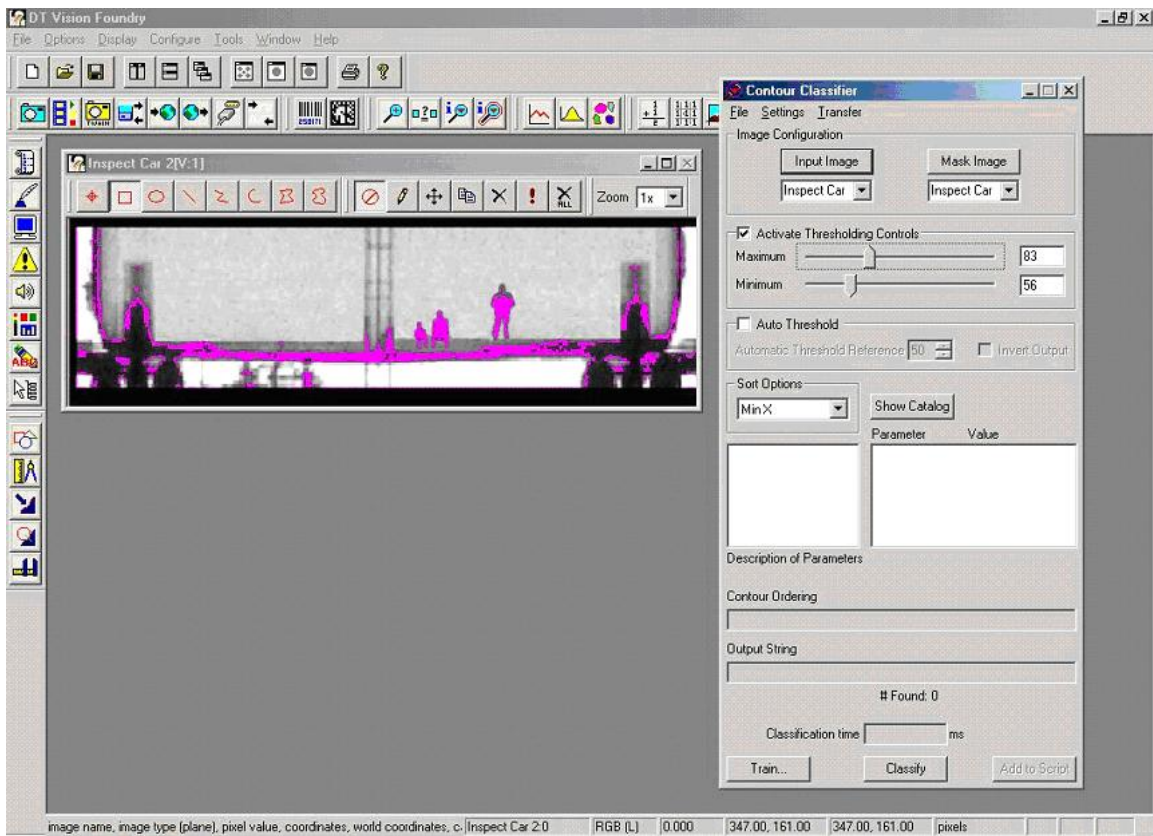


Figure 12: Contour Classifier

Inspection Technology: Final Summary Report

The AVIAN System consists of an industrial personal computer loaded with specially developed software, a touch-screen monitor and special sensors. The operator uses the touch-screen to select vehicle type prior to beginning the vehicle scan. The processed data provide the operator with a pass or search indication. The completed search can take as little as 14 seconds after the sensors are placed on the vehicle. The system has proven to be 100% effective and reliable in detecting human presence under ideal conditions (protection from moderate to high winds is required to eliminate false positive indications).

This device has proven to be 99.9% effective assuring a simple modeling effort to append human detection to the ASA based Seaport Inspection Process.

4 Seaport Inspection Process Demonstration

As previously described, a software demonstration simulating how the EDI manifest data, SCTS targeting algorithm and inspection technology hardware model operate and interact has been configured. How well such software components actually perform together to simulate a physical realization of the Seaport Inspection Process required a specific Software Test Plan (STP). Such a plan was produced and had as its primary thrust the verification of the software's ability to "learn" and hence improve its ability to detect suspect containers. A simple graphical user interface was generated for facilitating the software demonstration. Judicious use of the software demonstration should produce a plan for eventual physical demonstration at a major U.S. port.

The objective of the testing described in the Software Test Plan was to find the dependence upon SCTS prediction performance of the following variables: (i) number of containers opened in each simulated inspection, (ii) ratio of "suspect" random containers opened, and (iii) number of iterations the SCTS performs.

The number of containers inspected by the modeled equipment is to increase exponentially from two (2) to eight (8), a reasonable upper bound on the number of containers that could be inspected in a real world scenario. The percentage of random containers opened will increase monotonically from 10% to 50%, with a 10% increase on each test. Note, fractional percentages were always rounded up; therefore, at least one random package is always opened; thus producing redundancy in a number of the proposed tests. The number of iterations will increase monotonically from one (1) to an upper bound of fifty (50). Variable One has three different values while Variable Two has five different values. In total $3 \times 5 \times 50 = 750$ iterations should be executed. A subset of this testing was completed, because the complete set was not necessary to provide the essential results (described in Section 4.3).

4.1 Test Environment

There are a number of variables that were kept constant throughout testing; (i) the manifest data, (ii) the number of containers used to create an initial model for ASA, (iii) the size of the ASA model, i.e. number of variables used to model containers, and (iv) the length of time previous results are retained as part of the model. The values of each of these variables are described below.

To ensure a consistent environment during testing, the following factors were held constant:

Each test is dependent on the availability of EDI manifest data and the initial model of suspect containers. While the data may change through time, each test trial must use the same data. The introduction of new data may lead to additional rounds of inspection.

Inspection Technology: Final Summary Report

A major component of ASA is to learn from previous results. At initialization, no previous results exist and therefore container descriptions must be provided by the tester or supplied by the EDI data. Data parameters may be as minimal as one description or as fully specified as all available descriptions. For testing, a random initialization of ten descriptions was used.

The running time for ASA is dependent upon the number of variables used in the container model. As the number of variables increases, the model can capture more complex representations. In testing, set the number of variables in the model has been set to five.

ASA uses information about previous test cases to update its model of suspect containers. To facilitate learning it is often advantageous to reduce the importance placed on older results. In testing, eight iterations through ASA of the same historic data was determined to be the limit of usefulness.

4.2 Test Results

The results of the initial testing are based on a manifest with 128 containers. Of those 128, 15 were marked as highly suspect (.999) and the rest with the minimal suspect potential of 0.1. The results entail the specific learning characteristics and accuracy under various assumed inspection conditions and cannot be published at this time due to imposed national security regulations. The results of the tests that were performed are encouraging and there is no reason to think subsequent testing should differ.

4.3 Problems Encountered

Testing revealed a small number of limitations and constraints. The time required for an individual test was initially estimated to be about one minute, but is better estimated to be between 5 and 10 minutes.

One factor that increased execution time was the presence of EDI data fields defined by a large number of discrete categories. Because SCTS reasons about interactions between data fields, a large number of categories can have an exponential effect on the running time. Therefore, the testing was limited to data fields with 50 or fewer categories.

4.4 Future Testing

The recommended solution to the problem of increased execution time is to use an automated pre-processing step to aggregate the categories.

The Suspect Container Targeting System can be improved given more time to evaluate a number of additional scenarios. In addition, there are additional variables in the learning algorithm that can be modified in an attempt to improve the learning curve: (i) the number of data field interactions that determine the ASA model – as the number of data fields increases, the model can capture more complex representations; (ii) the length of time the results of previous iterations of SCTS are used in updating the current model; and (iii) testing on additional manifest data.