

2
AP

NAVAL POSTGRADUATE SCHOOL Monterey, California

S DTIC
ELECTE
JUL 27 1993
A **D**



AD-A267 128


THESIS

UNITED STATES NAVY SHIP EMPLOYMENT AND
CRISIS RESPONSE MODEL (ECRM)

by

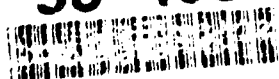
Lt. Takashi Roy Yamamoto
March, 1993

Thesis Advisor

Prof. Wayne P. Hughes

Approved for public release; distribution is unlimited.

93-16809



REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 1993		3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE United States Navy Employment and Crisis Response Model (ECRM)				5. FUNDING NUMBERS	
6. AUTHOR(S) YAMAMOTO, Takashi Roy					
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)				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) Since the collapse of the Soviet Union in 1990, the United States Navy finds itself searching for and defining new roles in the post-cold-war period. Design aids previously utilized to analyze the Soviet threat must be redesigned to assist in timely crisis response and the build-up of appropriate force levels for regional contingencies. Currently, no such model for crisis response exists for an operational commander and his staff that will help select the forces to respond. The Employment and Crisis Response Model, developed to assist in the contingency planning process, provides an indication of the total number of days required for the build-up of the desired force level on scene. The rapid calculation of this information allows decision makers to quickly analyze different response options when faced with a crisis.					
14. SUBJECT TERMS Crisis Response, Ship Employment, Cases, ECRM, Employment and Crisis Response Model				15. NUMBER OF PAGES 73	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL		

Approved for public release; distribution is unlimited.

**UNITED STATES NAVY SHIP EMPLOYMENT AND CRISIS RESPONSE
MODEL (ECRM)**

by

Takashi Roy Yamamoto

Lieutenant, United States Navy
B.S., United States Naval Academy, 1986

Submitted in partial fulfillment of the requirements for
the degree of

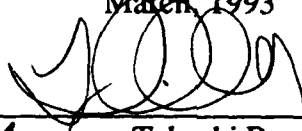
MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL

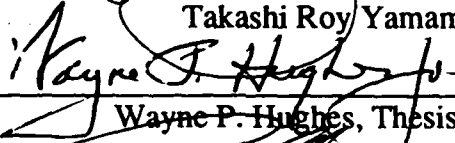
March, 1993

Author:

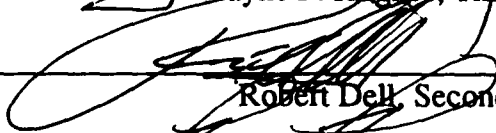


Takashi Roy Yamamoto

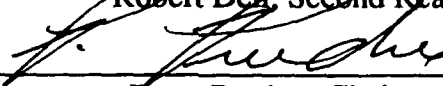
Approved by:



Wayne P. Hughes, Thesis Advisor



Robert Dell, Second Reader



**Peter Purdue, Chairman,
Department of Operations Research**

ABSTRACT

Since the collapse of the Soviet Union in 1990, the United States Navy finds itself searching for and defining new roles in the post-cold-war period. Design aids previously utilized to analyze the Soviet threat must be redesigned to assist in timely crisis response and the build-up of appropriate force levels for regional contingencies. Currently, no such model for crisis response exists for an operational commander and his staff that will help select the forces to respond. The Employment and Crisis Response Model, developed to assist in the contingency planning process, provides an indication of the total number of days required for the build-up of the desired force level on scene. The rapid calculation of this information allows decision makers to quickly analyze different response options when faced with a crisis.

Accession For	
NTIS	☒
DTIC	☐
Unannounced	☐
Justification	
By	
Distribution	
Availability Codes	
Dist	Avail and/or Special
A-1	

DISCLAIMER

The reader is cautioned that the computer program developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the program is free of computational and logical errors, it cannot be considered validated. Any application of this program without additional verification is at the risk of the user.

TABLE OF CONTENTS

I.	CHANGING TIMES AND NEW CHALLENGES.....	1
A.	THE WINDS OF CHANGE.....	2
B.	COMPUTERS AND CONTINGENCIES	3
1.	The Becker Model.....	4
2.	The Hall Model	5
3.	The Lindemann Model	6
4.	The Force Deployment and Estimation Model.....	7
C.	LIMITATIONS AND SOLUTIONS.....	8
II.	CASES.....	10
A.	THE BASICS.....	10
B.	CASES, FDOs AND CINCPACFLT.....	12
III.	US NAVY SHIP EMPLOYMENT AND CRISIS RESPONSE MODEL (ECRM)	13
A.	THE MODEL.....	13
B.	MEASURES OF EFFECTIVENESS AND MODEL OUTPUT.....	16
C.	AN EXAMPLE	17
IV.	PROGRAM.....	20
A.	DISCUSSION OF PROGRAM CODE	20
1.	Entering Ship Locations and Status.....	21
2.	Determining the Required FDO	22
3.	Dial's Algorithm	22
4.	Sorting the Ship List.....	26
5.	Generating Response Table and MOEs	26
V.	CONCLUSIONS	27
A.	LIMITATIONS OF ECRM	28
B.	FUTURE PLANS AND AREAS OF FURTHER STUDY.....	29

APPENDIX A.	UNITED STATES NAVY SHIP EMPLOYMENT AND CRISIS RESPONSE MODEL PROGRAM CODE.....	31
APPENDIX B.	GEOGRAPHIC LOCATION AND STATUS CODES	61
APPENDIX C.	CRISIS RESPONSE CODES.....	62
REFERENCES		63
BIBLIOGRAPHY.....		64
INITIAL DISTRIBUTION LIST.....		65

I. CHANGING TIMES AND NEW CHALLENGES

As the world evolves, so too does the National Security Strategy of the United States. The rapidity of change and the multi-polar threat requires a decision maker to make critical and timely decisions during crises. A tool to assist him would be of great benefit to meet any contingency or operational requirement. Currently there is no such tool available, therefore the need for such an instrument is evident. The ideal tool is an easy to use computer model that can be quickly incorporated into existing computer architecture.

The purpose of this thesis is to discuss the importance of a presence and crisis response model, and to provide the basic idea and programming logic to develop the model. A sample model, limited to aircraft carriers (CV) using hypothetical employment schedules is developed. Of particular interest is the determination of "optimal" employment strategies based on current Flexible Deterrent Options (FDOs) and available assets. Response options are based on calculation of response times, using a network shortest-path algorithm.

This chapter discusses the need for a crisis response model and examines four different forward presence and crisis response models currently being utilized. Model limitations are also discussed. Chapter II introduces the Capabilities Assessment and Evaluation System (CASES) and includes the purpose behind the framework, current models being utilized in CASES and finally, planned additions to CASES. Chapter III discusses the basic idea behind the United States Navy Employment and Crisis Response Model (ECRM) including the complexities of presence, objectives of crisis response, the goals, measures of effectiveness, and finally, utility. Chapter IV is devoted to discussion of the program code, including specific information pertaining to the basic model. Finally, Chapter V provides an overview of the model and

the model and concept, including limitations, extent of model testing, and areas of further study and future plans.

A. THE WINDS OF CHANGE

It is a time of great geo-political change. For the first time in the history of the United States, this nation stands alone as the only global superpower. The demise of the Soviet Union and Warsaw Pact in the early 1990s has, for the moment, decreased the likelihood of global warfare.¹ Although the nuclear threat continues to play a major role in political decision making, the dissolution of the Soviet Union has removed an enormous burden on the citizens and soldiers of the United States. Yet incidents in the Arabian Gulf in the 1980s and early 1990s are indicative of the relative instability of other regions of the world. Without the existence and direct influence of the Soviet Union, many nations or cultures may feel the need to address long suppressed external or internal objectives with military force, particularly during economically austere times.

At home, the United States faces the daunting task of cutting the budget to reduce the deficit of the 1980s. The need to balance the budget is clear, and defense dollars and the "peace dividend" make convenient sources of additional funds for legislatures intent on showing progress in correcting budgetary woes. In this climate of decreasing budgets, the United States must develop new strategies to deal with diverse international crises such as the invasion of Kuwait, humanitarian missions in Somalia and Bangladesh, and the bloodshed in the Balkan peninsula. This is not a simple undertaking, since the size of the Defense Department will be dramatically reduced. The future of naval strategy will hinge on flexibility and timely response to these

¹Sean O'Keefe, Frank B. Kelso, and Carl E. Mundy. ". . . From the Sea." U.S. Naval Institute *Proceedings*, November, 1992, p 93.

types of crises with limited assets. New computer models must be developed to assist planners in shaping strategy, preparing for unforeseen contingencies and maximizing force utility.

B. COMPUTERS AND CONTINGENCIES

Several computer models exist today to aid the planner in analyzing contingency operations, yet these models each address specific warfare mission areas (WMA). THOR provides data on strike warfare, Carrier Battle Force Logistics Expenditure and Resupply (CLEAR) models battle force logistical issues, and other models exist, or are being developed to address such WMAs as ASW (anti-submarine warfare), ASUW (anti-surface warfare) and amphibious warfare.² Most models were originally conceived and written to deal with the Soviet threat. Clearly, such scenarios have lost their urgency, if not their importance. These models do have utility in the era of Major Regional Crises (MRCs), Lesser Regional Crises (LRCs) and Low Intensity Conflicts (LICs), however, one important aspect is ignored by these models. Of greater importance to the decision maker and analyst during a crisis are questions pertaining to what forces are available to respond, and when those forces can arrive. A computer model which determines the location of available assets and calculates crisis response times allows alternative responses to be quickly generated and evaluated. Although computer models have been developed for various studies, none are adequate for assisting the decision maker in this process.

Several models that calculate forward presence and crisis response were examined for suitability. Two models were developed at the Center for Naval Analyses (CNA) to study forward presence and crisis response. Lieutenant

² Bruce Anderson and John Flynn. "CASES: A System for Assessing Naval Warfare Capability." Joint Directors of Labs C3I Symposium, 1990.

Commander Kevin Becker developed an APL (A Programming Language) model and John V. Hall developed a spreadsheet based model [Refs. 1 and 2]. Two other models were examined; another spreadsheet model developed by Michael J. Indemann, written for a study conducted at the Naval Surface Warfare Center (NSWC), and one model developed for the Naval War College (NWC)/NSWC Naval Surface Warfare 2030 Symposium held at NWC in December 1991 [Refs. 3 and 4]. All four models are discussed briefly below.

1. The Becker Model

Becker developed a presence and crisis response model, written in APL, for a study conducted at CNA for the Commander-in-Chief, Pacific Fleet (CINCPACFLT). This model calculates the presence and crisis response capabilities of Pacific-based aircraft carriers. Becker devised an algorithm, using the Pacific-based CVs, to examine their ten year deployment and maintenance cycles. It then calculates the expected presence of carriers in the Pacific for the next ten years. The resulting data is displayed as a histogram depicting the percentage of time various numbers of CV/CVBGs (aircraft carrier battle groups) may be present in a geographic region over time. Although Becker uses Pacific-based CVs for his analysis, any mix of surface platforms and submarines can be used in the model as long as the necessary historical employment data is available. Based on this presence data, he utilizes a transportation problem to determine estimated response times for various crises and CV peacetime deployment options. [Ref. 1]

Since the model is primarily based on historical data, several weaknesses are inherent from the perspective of the theater level decision maker. Several assumptions, as stated by Becker, are necessary to determine the estimated presence of CVBGs. Two primary areas of difficulty are that CV

schedules are assumed to be recurrent over the ten year period examined, and "make-ready-to-sail" times have been simplified.³

Although CV schedules are based on some basic cycle plan, it is easy to understand that CV employment cycles are neither symmetric nor recurrent. Clearly, the last ten years of CV employments will not likely be an effective example for the next ten years.⁴ The CV has been the favorite tool for crisis response, and will continue to play a major role in diplomacy, however, anticipated reductions in the CV force will make it difficult to model future CV presence from historical CV data. In summary, Becker's model cannot (nor was it intended to) be utilized as a tool for commanders in an operational environment. Another drawback for the decision maker is that the algorithm is written in APL, making modifications difficult for most users. The model's strength lies in its ability to provide rapid "what-if" information for long-term general strategic planning and budget studies.

2. The Hall Model

Hall developed a desktop computer spreadsheet based presence and crisis response model at CNA. In it are even greater assumptions and simplifications.⁵ The Hall model assumes fixed deployment schedules and

³ Make-ready-to-sail time is the number of days required to make a CV that is in some type of maintenance availability deployable again.

⁴ Several perturbations in the deployment patterns of CVs occurred in the last decade including responses to actions by Libya and Iraq. Although it can be argued that crises will arise in the future, it appears unlikely that CVs will be deployed again on the scale required to support Desert Shield and Desert Storm in the near future.

⁵ While assigned to CINCPACFLT, CAPT William Schwabe, USNR, revised Hall's model to reflect different assumptions and to model the problem more realistically.

steady state conditions. Using average values, the model is able to calculate how much presence forces can generate in some geographic region, and how rapidly they could respond to crises. Again, this model provides data concerning average forward presence and average crisis response times, but is not intended to be utilized by the theater level commander in an operational environment in which position and readiness status are known at "go time." It is intended to provide quick and easy answers to "what-if" questions concerning alternative deployment schemes such as the lengthening of deployments, alterations in maintenance schedules, forward basing of carriers, or changes in OPTEMPO (operational tempo) requirements, etc. Although useful to the Pentagon budget process and long-term strategic planners, neither serves the purpose of executing contingency operations. [Ref. 2]

3. The Lindemann Model

The Lindemann peacetime deployment model, similar in some respects to the Hall model, calculates an estimate of total surface force structure, by ship class, that is required to provide a specific worldwide presence. There are two versions of Lindemann's model available, both of which were written for the Apple Macintosh™ computer using Microsoft Excel™. Version 1 calculates forward presence required based on geo-political issues. Since the primary purpose of this thesis involves crisis response, this version was not considered for evaluation. Version 2 allows the user to directly input presence requirements. This version also calculates various costs associated with the force level requirements necessary, and more importantly from the stand point of this thesis, also calculates expected CV surge capability. Unlike the previous two models, Lindemann does not limit his analysis to CVs, instead, distinguishing between CVBGs, ARGs (Amphibious Ready Groups), SAGs (Surface Action Groups) and CLGs

(Combat Logistics Groups). He also analyzes ship types, such as CV, cruisers, destroyers, submarines and logistical support ships. Only his analysis of surge capability is limited to CVs. Nevertheless, this model suffers the same limits that the previous two models exhibit in that the results are expected values and averages. [Ref. 3]

4. The Force Deployment and Estimation Model

During the Naval Surface Warfare 2030 Symposium II, held at the Naval War College in Newport, Rhode Island, in December 1991, similar issues of presence and response were examined. Of particular interest is the Force Deployment Estimation Model (FDEM) developed for the symposium. FDEM allows users to:

- Explore alternative deployment concepts and innovative force packages;
- Study force structure effectiveness; and
- Study trade-offs between alternate naval force deployment concepts.⁶

Based on a specified force structure and world events, the model provides as output, a proposed allocation of forces. The inputs to FDEM can be adjusted to study hypothetical force structures, force packaging plans, basing configurations and demands for naval forces. As with the Hall and Lindemann models, however, this model represents a steady state response. FDEM addressed similar issues studied and analyzed by the Lindemann model, and may be the result of the efforts by NSWC to answer equivalent questions. [Ref. 4]

⁶ *Game Book*. Naval Surface Warfare 2030 Symposium II, Newport, 3-5 December 1991, p. V-1.

C. LIMITATIONS AND SOLUTIONS

Each forward presence model discussed has strengths and applications, but none are designed for the theater level commander in analyzing options for specific contingency operations with specified requirements for force presence. The ECRM (Employment and Crisis Response Model) is developed in this thesis to assist the theater level commander.

In ECRM, existing ship quarterly and cycle schedules should be used to provide real, timely and accurate data for the program. The drawback to this method is that conversion of employment schedules into a format suitable for computer calculation will be time consuming. Nevertheless, for contingency planning the best inputs must be used in providing options for planners and decision makers.

In the current milieu of flexible crisis response, earlier stand-alone analytical warfare mission area (WMA) models, such as THOR, are too cumbersome to be of any use to the planner who must make critical and timely decisions. Other models such as the Becker and Hall models discussed earlier, are simply not applicable. The rapid development of computer technology, however, has enabled programmers to integrate individual WMAs into one package.

One such package under evaluation is the Capabilities Assessment, Simulation and Evaluation System (CASES) now in use at CINCPACFLT.⁷ There are, however, notable shortcomings; CASES models campaigns by assuming that assets are on station and available for operations. What if there are no assets available in theater? What are the available options to the decision maker? How soon can the planner expect to be able to conduct

⁷Bruce Anderson and John Flynn.

military operations in support of diplomatic and economic efforts? These crucial questions that are not answered by CASES.

The importance of such questions are obvious in a multi-polar world. The draw down of military forces will lead to alternative deployment schemes by the Navy to meet global commitments. The planner must know the location and readiness of his forces , and when a crisis develops, he must be able to quickly determine the means of response with a preferred course of action. The need for a model to provide planners with just such options is evident in this remark by Dr. Ray Runyan, Senior Analyst at CINCPACFLT,

A major initial requirement for timely and effective crisis response is the build-up of forces to appropriate levels, i.e., what forces are available, where are they, what is their status, when can they be on station, etc. CASES has much of this information now or has access, but a planning algorithm to bring it all together for the planner is needed.⁸

To address the problem identified by Dr. Runyan, a model to assist in the planning phase of any contingency is to be developed. The United States Navy Employment and Crisis Response Model (ECRM) will show how to solve the problem posed and provide a valuable tool to assist the decision maker. Ultimately, the model can be integrated into the CASES framework to allow maximum flexibility and speed for the planner.

⁸ Ray Runyan. " FPC Models Board." Models Review Board, San Diego, 16 June 1992.

II. CASES

CASES originated as a battle management application in DARPA's (Defense Advanced Research Projects Agency) Strategic Computing Program to demonstrate distributed heterogeneous processing, parallel processing, and knowledge-base support. It is a decision-support system used by CINCPACFLT to support the development analysis and execution of war plans and contingency operations.⁹

A. THE BASICS

In essence, CASES is a classic campaign analysis tool that is utilized in assessing warfighting capabilities on a theater level. In the past, campaign analysis was primarily conducted by operations analysts who modeled each separate warfare mission areas (WMA) as a batch computer process. Such an undertaking was formidable due to the tremendous amount of information that was required to be assembled, modeled in detail, and evaluated. Analysts were required to prepare model inputs, interpret model outputs, apply statistical principles and present their findings from multiple computer runs. Normally such a task required weeks or even months to accomplish. CASES automates the assessment process through an interactive workstation to perform analysis in hours or days.¹⁰

Standard Navy models are integrated within the CASES interactive framework which allows the user, typically military operators rather than analysts, to set up and visualize warfare plans and scenarios. This was

⁹ Bruce Anderson and John Flynn.

¹⁰ Ibid.

accomplished by taking advantage of recent developments in GUI (graphical-user-interface) technology, as well as increased capabilities of workstations and local-area management systems and networks. Through a series of windows and selectable options, the user can access the huge database and create campaigns at the theater level, or simply analyze individual models within the CASES framework. With this capability, the operator can readily compare various contingency scenarios for the decision maker in minutes or hours rather than days or weeks. During the initial action and subsequent contingency planning phases, this capability is fairly clear.

CASES release 6.2 is the latest model available and includes the following WMA models within its architecture:

- ASW Environment (RAYMODE, PE, GEM-DB, etc.);
- ASW Sensor Capabilities (Sweep/Search);
- Campaign-level ASW (NACM);
- CVBF Defense (ASBAT);
- Air Threat Assessment;
- Strike (THOR); and
- CVBF Logistics Expenditures And Resupply (CLEAR).

Additional models are planned for inclusion into the CASES environment. Those models are:

- War at sea (WASPS);
- Amphibious Warfare;
- Shallow Water ASW; and
- Land Campaign.

B. CASES, FDOS AND CINCPACFLT

CINCPACFLT has begun to examine the concept of force packages as a means to simplify contingency planning and response. A classified message from CINCPACFLT contains detailed FDO (Flexible Deterrent Option) packages for various crises, devised from historical records involving force employment as well as diplomatic and economic actions taken by the United States in the past. The FDOs available to decision makers run the gamut from simple diplomatic actions, through various economic and humanitarian actions, to complete wartime force packages. Although general in nature, these FDOs are specific enough to allow the theater-level commander to deploy necessary forces as an initial response.

Utilizing the FDO concept, it becomes apparent that combining FDOs with a crisis response model will allow the theater-level commander to examine available assets and select the best package to respond to the contingency. He will also have the ability to compare different FDO packages, if he chooses, by making simple user inputs and comparing the data.

Enclosing this within the CASES framework increases the analytical capabilities further while maintaining simplicity. Not only will the decision maker be able to compare the FDO packages available and deploy the most suitable forces in the shortest amount of time, he will also be able to have CASES provide the necessary analytical data to examine, amongst other options, the strike, ASW or ASUW capabilities of the various FDOs suggested by the model, or input by the operator.

III. US NAVY SHIP EMPLOYMENT AND CRISIS RESPONSE MODEL (ECRM)

Four critical mission areas have been defined by ex-President Bush in shaping national strategy for the future [Ref. 5] and are echoed in the Secretary of the Navy's White Paper, . . . *From the Sea*. Those four areas are strategic deterrence and defense, forward presence, crisis response and reconstitution.¹¹ The requirement for a crisis response model is clear, and CASES provides an excellent framework to incorporate such a model. ECRM addresses the issues of forward presence and crisis response, making the model a valuable asset and tool for the planner and decision maker.

A. THE MODEL

The objective of ECRM is to provide the decision maker such as CINCPACFLT with a valuable tool in planning for contingency operations in his theater of operations. The model determines all assets available to the planner, and from within a group of eligible candidates, selects the best choice of ships to respond to the crisis. The response is dependent on the nature of the crisis, the level required, and the CINCs choice of FDOs. The model is intended to be implemented with all major U.S. Navy surface combatants in the Pacific, however, the scope of the thesis is limited to the employment and response of CVs so that the model can be easily followed. The concepts and ideas demonstrated here can be extended to include all major ship types such as amphibious warfare assets as well as cruiser/destroyer type ships. Figure 1 provides a flow diagram of the basic operation of the model.

¹¹ Stan Weeks. "Crafting a New Maritime Strategy." U.S. Naval Institute *Proceedings*. Jan 1992. p. 32.

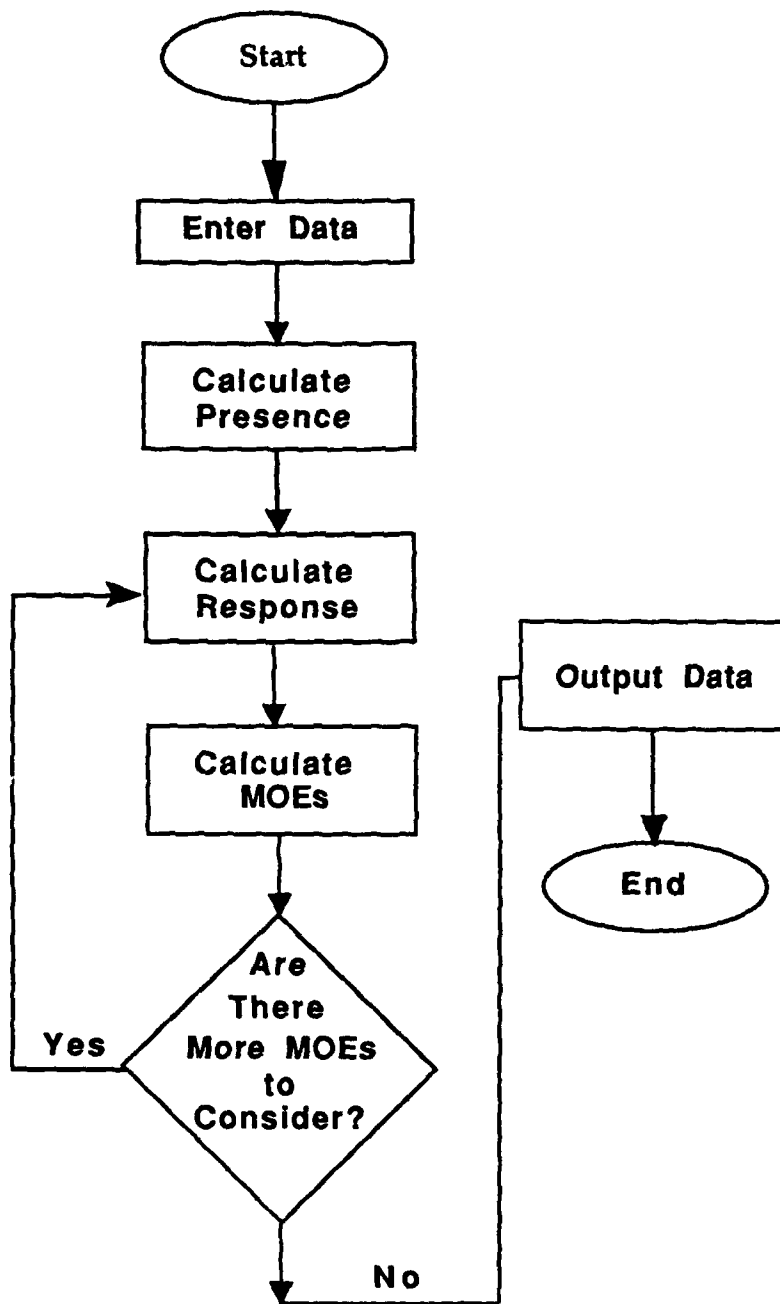


Figure 1: Model Flow

When a crisis develops, the model operates with the following data:

- Ship schedule database;
- Region of interest;
- Nature of the crisis;
- Speed of advance (SOA);
- Possible port stops; and
- FDOs.

With this data, the model obtains a "snap-shot" of the current situation which serve as the initial conditions of the model.¹²

Next, the model selects several possible deployment and response FDOs. As an example, suppose that a typhoon devastates the island of Maui. The CINCPACFLT user model determines that the crisis requires the disaster relief or humanitarian mission FDOs. From within this sub-group, the extent of the required assistance is determined, based on information provided by the user. An austere option could include sending a single destroyer to the island to provide medical and communications assistance as well as some limited fresh water and emergency power requirements. If the crisis is of greater severity, the next option would be sending a task unit to the area, centered on a major amphibious vessel with LCAC capability, as well as Navy construction battalion (CB) detachments. For this thesis, the required force will be the number of CVs required to be on station. Once the FDO is selected, the algorithm examines all available assets. Initially, the algorithm sorts the assets by status, i.e., whether the CV is conducting operations nearby, is

¹² For the purposes of this thesis, a hypothetical schedule based on ship homeport and specific deployment regions was utilized as the data for the model. It can be assumed, easily enough, that the data originated from schedules provided by the various fleet scheduling conferences.

currently in an extended maintenance availability or simply returning from a deployment. From within this sorted group, the algorithm selects the CV(s) with the shortest response time. The shortest response times include normal port stops scheduled for replenishment purposes and any necessary make-ready times. The initial output display is in tabular form so that the decision maker or aide can rapidly determine if the CVs selected are in fact, the best options.

Once the desired FDO is selected, the model chooses the best assets to respond, calculates the required measures of effectiveness (MOE) and outputs the data to the user. The output data consists of the "optimal" FDO with the required MOEs. In addition, the model also provides data for FDOs that are one level above and below the recommended package to provide the decision maker with additional options.

B. MEASURES OF EFFECTIVENESS AND MODEL OUTPUT

There are two primary measures of effectiveness which are used to assess the crisis response requirements:

- Time delay for the complete FDO to arrive at the scene.
- Amount of combat potential or humanitarian assistance level available in the region each day during the buildup.

The output of the program provides information to the user on the amount of combat potential/humanitarian assistance available as a ratio of numbers on station each day versus the user specified total combat potential or assistance level required to meet the crisis. With this information, the user can readily determine when all required units of the FDO will be on station, and observe the number of days required to reach this level.

C. AN EXAMPLE

To provide an example on how the model runs, suppose a crisis develops in the Philippine Islands. A coup d'etat has toppled the democratically elected government. At the moment, CINCPACFLT has six CVs that are assigned to the Third and Seventh Fleets. CV1 is in an upkeep period in Alameda. CV2 is on station in the Indian Ocean, monitoring activities in the Arabian Gulf. CV3 is nearing the completion of an upkeep period in San Diego. CV4 is currently undergoing an overhaul in Washington and is not expected to be available for more than 30 days. CV5 is underway conducting training operations in Southern California. CV6 is underway conducting local operations near Japan. Based on the information obtained, the model outputs Table 1.

TABLE 1: FDO ALLOCATION

FDO Allocation					
CV#	Status	Location	Distance	Stops	Est. Days
6	1	3	1689	0	3
5	1	1	6358	1	15
3	2	1	6358	1	16
1	2	1	6358	1	45
2	4	6	4313	0	30
4	5	1	6358	1	345
Note: Estimated Days include standard logistics stops and make-ready					

From the table, CV6 appears to be the best option, since it is already at sea. Assuming an average transit speed of 18 knots, CV6 will require 3 transit days to arrive on station. Since the FDO only requires a single CV, CV6 would be the "best" choice. The next choice would be CV5, underway in Southern California. With a 14 day transit, plus one day for supply on-loads in Pearl Harbor, she could make it on station in 15 days. CV3 is nearing the end of an upkeep period with only one day remaining. With transit time and stops included, she would require 16 days. CV1 is also in an upkeep period, but would require more time to prepare for a deployment. CV2 is not permitted to leave the Indian Ocean, therefore, her estimated response is expected to be greater than 30 days. CV4 cannot even be considered for the response, since completion of her availability would require more than 30 days.

At this point, the decision maker, CINCPACFLT, has the option of changing the recommended order of response priority. Although the FDO package does not require a second CV, she decides that another CV would be prudent. She decides that the model solution is not the best choice because of the excessive amount of time required for the second CV to respond to the crisis. Instead, she consults with her staff and decides that CV2 will be the second CV to respond. Orders are dispatched to CV2 ordering her to leave the Indian Ocean and proceed to the Philippines.

The recommended priority is then adjusted to reflect the new FDO allocation solution, and the analysis begins. The model calculates the percentage of the FDO to arrive on station from D-day and outputs the time delay required for the entire FDO to arrive on station. In addition, the model also outputs to an external text file, the cumulative percentage of the FDO arriving on station each day. This data can then be displayed graphically using a separate commercially available graphics generating computer program such as *DeltaGraph Professional*® or any good spreadsheet program. The

output of the example problem is provided as Figure 2, graphed with DeltaPoint's software *DeltaGraph Professional*® [Ref. 6].

The model selects the next lower FDO, in this case, no CVs (ultimately this would imply a SAG, but one CV is the default) and repeats the calculations. Then, the model selects the next higher option, in this case, two CVs, and completes the analysis.

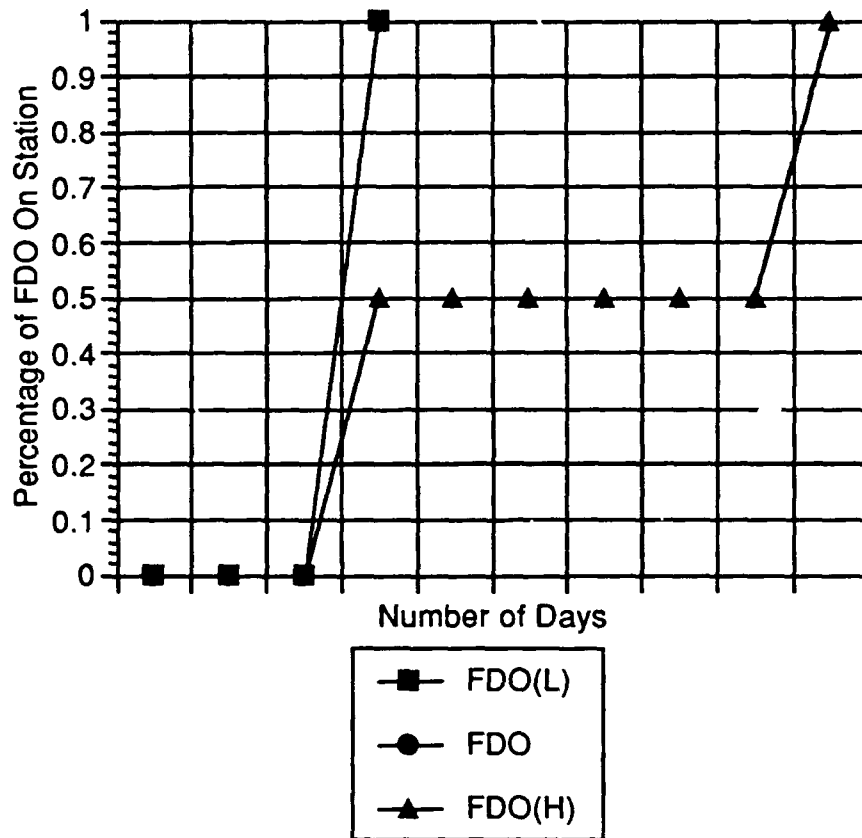


Figure 2: Graphical Display of Percentage of FDO Arriving on Station.

IV. PROGRAM

The goal was to write the computer code, included as Appendix A, in a manner that would be easily transportable to the UNIX environment, or even the PC desktop environment.¹³ This necessitated the use of ANSI (American National Standards Institute) conforming C computer code. Since the C programming language is fairly prevalent and popular amongst current programmers, and the code conforms to ANSI C, the code can be easily modified for any future use for the model. This allows ECRM to be incorporated into existing theater level campaign analysis models used by the Navy or possibly into campaign analysis models used by sister services.

A. DISCUSSION OF THE PROGRAM CODE

Before discussion of the actual computer program can begin, a brief introduction to the data structures is provided. The main data structures utilized in the program are ship type, nodeSet type and edge type structures. The most important of these structures, the ship type, is used in the ship response priority and represents individual units, in this case, CVs. Later in this chapter, the description of the ship response priority, stored as a doubly-linked list, will explain the functions of the fields in the ship type structure. The ship type structure is made up of the following fields:

- ID unit identification;

¹³ The reader is cautioned that the computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logical errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

•location	location of unit;
•status	current unit status;
•cP	combat potential;
•hV	humanitarian value;
•distance	distance, in nautical miles, to the crisis;
•stops	number of port stops normally required;
•days	number of days needed to respond;
•pNext	pointer to next unit; and
•pBack	pointer to preceding unit.

Although the combat potential and humanitarian value fields are available, those values were assumed to be uniform and equal (1.0) for each individual notional CV examined in this thesis. In reality, the values vary with ship class and type, and in some cases, individual ships within a class, depending on the capabilities inherent in specific units. In the future, quantification of both values will be required for calculation of the second MOE once all surface ships are implemented into the model.

1. Entering Ship Locations and Status

Following the initialization of variables and arrays, the main segment calls a sub-routine to read in the location data of assets in the theater. The external data file used is stored as an ASCII (American Standard Code for Information Exchange) text file in matrix form.¹⁴ Each row of this matrix represents an individual unit and the columns represent time periods. Each time period is assumed to be two weeks in length, therefore, the entire matrix of unit locations represents approximately one quarter of the yearly schedule.

¹⁴ Commercially available software such as spreadsheet programs can be used to enter or modify this data, and all other external file data, as necessary.

The program reads in each location, assigning a memory location for each entry. As each location is read in from the external file, the program builds a location array in memory similar to the external data matrix.

Once the location information is entered, the computer then opens two files, one containing ship status files, and the other containing estimated make-ready times, in days. As the computer reads in the ship status code, memory large enough to hold the ship type data structure is allocated for each individual ship. When memory is allocated, the computer enters the individual unit status into the ship status field. The make-ready time is then determined for the individual unit and stored in the ship field representing days. A description of location and status codes is provided as Appendix B.

2. Determining the Required FDO

The program queries the user concerning the crisis code and source of the crisis. The crisis codes utilized in this model are provided as Appendix C. When the crisis code is entered by the user, the program calls the FDO function to determine the standard FDO required for the crisis. The integer value that is returned is used to determine the number of particular ship types required to respond to the crisis. For this thesis, the ten's digit location indicates the number of CVs required, and the one's digit indicates the number of LHXs (LHA - amphibious assault ship, general-purpose; LHD - amphibious assault ship, multi-purpose) required. Again, the program only examines the CVs required although the FDO code that is returned can be easily adjusted in the future to reflect other platform types, such as cruisers and destroyers. The program then queries the user as to the location of the crisis.

3. Dial's Algorithm

Given the location and status of available assets and the crisis region of interest, rapid response times are a simple function of distance and speed,

as taught in basic algebra. If speed is fixed, then transit time is a linear function of distance, and the response time can be solved as a single-source, shortest-path network problem. It will be assumed that all assets will respond at a standard fixed SOA (speed of advance), assumed to be 18 knots. In the future, it may be necessary to incorporate variable speeds into the model if air assets are to be included.

Based on the location of ships and the location of the crisis, the algorithm uses a network model to determine shortest-path distances between the regions of interest. The location of assets, the crisis region and way stops can be defined as nodes in a graph. The weights assigned to the edges of the graph are the geographic distances between the nodes, determined by DMA (Defense Mapping Agency) and provided in Table 2 below. Using such distances, a graph can be created such as the one depicted in Figure 3 below. Although seemingly simple, the problem of shortest-paths rapidly increases in complexity with increasing numbers of nodes, which can be added in the future to reflect various geographic regions. It is anticipated that no more than 15 nodes would be required to provide sufficient resolution to encompass the Pacific theater of operations.

Several algorithms such as Dijkstra, label-correcting, Bellman-Ford, and Ahuja-Mehlhorn-Orlin-Tarjan are available to solve the shortest-path algorithm. In selecting a method to solve the problem, it is clear that a method for solving a graph with nonnegative weights and no negative weight cycles would be a sufficient algorithm, since geographic distances are never nonnegative. Dial's shortest-path algorithm, a modified version of Dijkstra's algorithm, was utilized to calculate the shortest-path distance with the crisis region serving as the source node. Dial's algorithm, as with Dijkstra's algorithm, requires that all edges of the graph have nonnegative weights, and the graph must not have negative-weight cycles. The main element of Dial's algorithm utilizes a relaxation technique. In short,

relaxation is a method that repeatedly decreases an upper bound on the actual shortest-path weight of each vertex until the upper bound equals the shortest-path weight.¹⁵

No comparison of the various methods was conducted. A run time analysis of the various algorithms was not investigated under the assumption that the number of nodes would not significantly impact the results. Though Dial's algorithm was utilized here, any of the other methods available to solve shortest-path problems could be implemented and used.

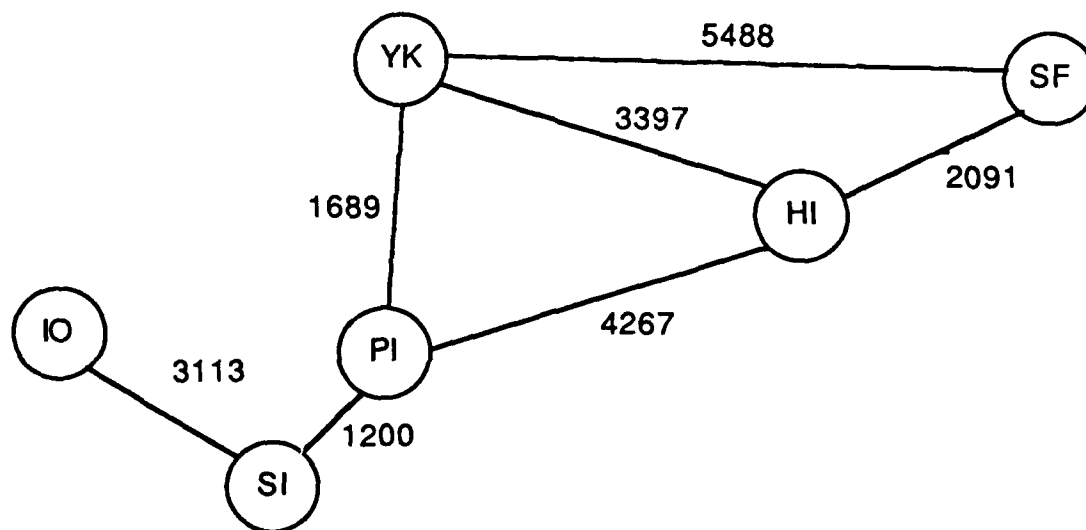


FIGURE 3: A Simple Transportation Network

Based on the distance solution obtained from Dial, the program must update the distance field for each unit. For each ship, ECRM compares the ship's present location to the location of the first node in the distance sequence. If the locations match, the ship data structure is updated by entering the distance in the ship distance field, and calculating the number of days

¹⁵ Thomas H. Corman, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*. (Cambridge: The MIT Press, 1990), p. 520.

required to respond to the crisis. Response days are dependent on the ship's status. If the status code indicates that the ship is unavailable to respond, the number of days is set to thirty, the default for units not able to respond. If the ship is able to respond, the number of transit days is calculated by dividing the total distance by 24 (hours per day) and again by the crisis transit speed of 18 knots. The number of days required for logistical stops while transiting to the crisis is added to produce the total steaming days required to respond. This value is then added to make-ready time to arrive at the estimated total number of days required to respond.

TABLE 2: DISTANCES BETWEEN NODES

Travel Distances between nodes						
	SF	HI	YK	PI	SI	IO
SF	0	2091	5488	6358	7558	10671
HI	2091	0	3397	4267	5467	8580
YK	5488	3397	0	1689	2889	6002
PI	6358	4267	1689	0	1200	4313
SI	7558	5467	2889	1200	0	3113
IO	10671	8580	6002	4313	3113	0
Defense Mapping Agency chart, <i>The World</i> , edition 1-DMATC, stock no.						

SF - San Francisco/West Coast of United States

HI - Hawaii/Mid-Pacific

YK - Yokosuka, Japan/Northwest Pacific

PI - Philippine Islands/Southwest Pacific

SI - Singapore

IO - Indian Ocean

4. Sorting the Ship Priority List

Following the calculation of total response times for each ship, the ships are then organized into a ship priority linked-list. Before the ship priority list can be displayed for the user, it must be sorted by status and distance. The ship list is reorganized as a binary tree array where the first node in the array represents the ship with the highest status value. This value is then exchanged with the last ship in the list, and the heap sort algorithm is called, recursively if necessary, to sort the new list. Again the first node is exchanged with the last node, and the process is repeated until all units have been sorted. The program then compares the status values in the ship list. Ships with equal status values are then sorted linearly to prioritize the listing by status and distance.

5. Generating Response Table and MOEs

The ship priority list is then output in tabular form, as in Table 1, so that the user can examine the response recommendations and make changes as necessary. The user has the ability to change the order of the response, the status of any ship, the number of days in port for logistical purposes, the make-ready time and the default transit speed. When changes have been entered, the ship priority list and table is updated, reflecting the new information. When satisfied with the resulting list, the user can resume the model, allowing for calculation of the measures of effectiveness.

V. CONCLUSIONS

Recent upheaval of the global geo-political climate, combined with the capricious nature of the Earth, regularly demonstrate that crises around the world can occur suddenly and unexpectedly. In an era of diminishing budgets and downsizing, the need to respond to crises has not abated. In fact, it can be argued that with the United States as the sole superpower, our global responsibilities have increased. The United States Navy must be ready to meet these demands quickly and efficiently to ensure the restoration of security and peace.

To assist the decision maker in determining an adequate response to crises, a decision aid that can analyze various FDOs and deployments with a campaign analysis model such as CASES is invaluable. Currently, no such deployment model exists to assist the planner and decision maker.

Four presence and crisis response models were evaluated and determined to be unsuitable for analyzing actual contingencies. Therefore, a need exists for a contingency planning model that can be incorporated into the CASES framework. The developed Employment and Crisis Response Model is the solution. ECRM allows computer operators and watchstanders, rather than the analysts, to quickly determine courses of action in crisis response.

One of the model's strengths is its flexibility, allowing operators to examine the best available forces to meet a required FDO recommended by the model, based on Operational Support System (OSS) data for current locations of CVs and normal transit routes. If the model solution is not satisfactory, the user can alter the model solution prior to calculation of MOEs to meet the needs and desires of the decision maker in choosing the "best" force. This flexibility allows users to compare different deployment options

quickly and provide prompt answers to questions concerning alternative deployment options.

ECRM is an adaptable model, since the program was written in ANSI C. This language also insures compatibility with CASES in the CINCPACFLT OSS. In addition, the computer code follows a modular design, allowing for easy enhancement of the program. Other service assets, such as Army divisions or Air Force squadrons can be easily programmed and added as a module by following the design of this thesis prototype. Although intended to serve as an input into CASES scenario analysis, ECRM also has the ability to stand alone on any desktop or notebook computer.

With appropriate data from OSS and some modifications to the data entry interface, the CV model can be implemented at CINCPACFLT FPC as is.

A. LIMITATIONS OF ECRM

Although ECRM is definitely a push in the right direction, it is still a long way from providing the necessary capabilities essential in analyzing scenarios. Currently, ECRM is limited to calculating the response times of aircraft carriers. Useful as that may be, clearly it does not provide the necessary flexibility in describing full response capabilities, including FDOs made up of amphibious platforms, or surface action groups.

ECRM does not determine the preferred FDO in a crisis. Simply put, the model does not examine and compare the various combat and humanitarian potentials of different ships and aircraft combinations to respond to crises. Doing so would probably necessitate the use of an integer programming model to obtain a solution. This does not, however, limit the usefulness of ECRM. The FDO concept was developed to allow decision makers to "plug-in" deployment modules that meet specific crisis criteria. These deployment packages greatly simplify the task of the decision maker, providing an

excellent basis for more detailed analysis of additional options. Once the FDO is selected, the model recommends the resources that will meet the requirements of the FDO *in the shortest amount of time*, based on the data and actual states of CINCPACFLT units.

ECRM does not have the ability to process more than one crisis at a time. Nevertheless, the model has the flexibility to compensate. In most cases, the discrete nature of crises will allow allocation of forces for one crisis, followed by allocation to another. Analysis of one crisis will update the status markers and database, which will be reflected in the analyses of subsequent crises. Prioritizing crises prior to model input can greatly enhance the quality of the model output, as is expected.

ECRM is not a substitute for thorough analysis conducted by expert analysts—their value in providing studies and modelling cannot be replaced by ECRM. ECRM only serves as a catalyst for rapid decision making when faced with a crisis requiring instantaneous action.

In summary, the two main limitations to ECRM include the inability to determine “optimal” FDOs in a crisis, and the inability to handle simultaneous crises, however, the model does provide the necessary flexibility to allow users to minimize the effects of these limitations.

B. FUTURE PLANS AND AREAS OF FURTHER STUDY

This thesis does not develop an all encompassing CASES module that can be easily integrated into the framework immediately. The purpose of this thesis was to develop the basic model prototype required to program such an ultimate model, employing the ideas presented here. The goal of the model, as the title suggests, is to incorporate all major U.S. Navy surface combatants available to the decision maker and planner. By contrast, the prolific use of mines as a cheap substitute for capital ships has been demonstrated on

numerous occasions. It may be wise to include mine assets in a crisis response model.

The eventual inclusion of combat logistics assets is recommended. Timely response of combat potential or humanitarian assistance is vital in a crisis, but immediate response is only a portion of the whole effort. The sustainability of FDOs in a crisis region with fuel, ammunition and supplies is of extreme importance both to the planner and decision maker. An integer programming algorithm or other method could be used to calculate the logistical requirements of each FDO, and then select the logistical assets required to sustain the FDO for some period of time is an important next step. CASES will have the ability to evaluate logistical concerns once the CLEAR (CVBF Logistics Expenditures and Resupply) model is fully integrated into the CASES environment. A means to tap into this model should be investigated.

Modules for other Navy assets such as submarines, aircraft squadrons and construction battalions should be developed to enhance the capability of the model. To meet joint operational requirements, ECRM should include major Air Force and Army assets, such as light infantry or airborne divisions and rapid response tactical air lift. The drawdown in the Department of Defense will only increase the need for joint operations in crisis response. Modules for the sister services will be beneficial to the joint task force commander as well as the theater level decision maker.

Finally, ECRM should incorporate a forward presence algorithm. Such an algorithm will allow planners and decision makers to assess the impact that certain deployment packages will have on future forward presence and short-term deployments. This "what-if" forecasting capability may provide additional constraints on selection of assets that make up the recommended FDO.

APPENDIX A

```

/*****
*   Structs Header Module
*   Written by LT Takashi R. Yamamoto
*   Operations Research Department, Naval Postgraduate School
*
*   Contains all struct types defined, as well as #define
*   assignments
*****/

#define      maxNodes      6
#define      numCV         6
#define      speed         18
#define      infinity      INT_MAX
#define      maxEdge       4268

struct Posit
{
    short  period;
    short  location;
    struct Posit *next;
};

struct ship
{
    short  ID;
    short  location;
    short  status;
    float  cP;           /* measure of combat potential */
    float  hV;           /* measure of humanitarian value */
    short  distance;
    short  stops;
    short  days;
    struct ship *pNext;
    struct ship *pBack;
};

struct nodeSet
{
    short  node;
    short  distance;
    struct nodeSet *pNext;
};

```

```
    struct nodeSet    *pBack;  
    short    bucket;  
};
```

```
struct pNode  
{  
    struct nodeSet    *pTr;  
};
```

```
struct edge  
{  
    short    nodeOut;  
    short    nodeIn;  
    long    distance;  
    struct edge    *pNext;  
};
```

```
struct node  
{  
    struct edge    *pAdjNodes;  
};
```

```

/*****
*      Global Variable File
*      Written by LT Takashi R. Yamamoto
*      Operations Research Department, Naval Postgraduate School
*
*      Global declarations for ECRM.
*****/

```

```

#include      "structs.h"

```

```

extern FILE      *data;
extern short     source, D, i, crisis, region, xsitSpeed;
extern short     fDO;
extern Boolean   good;
extern struct    node   adjList[ maxNodes ];
extern struct    pNode  B[ maxEdge + 1 ];
extern struct    nodeSet *P[ maxNodes + 1 ];
extern struct    nodeSet *pFirst, *pLast, *setS, *sLast, *sMarker;
extern struct    nodeSet *pVertex;
extern struct    Posit   CVLocation[ numCV ];
extern struct    ship    *CV[ numCV ];

```

```

/*****
*      Crisis Code File
*      Written by LT Takashi R. Yamamoto
*      Operations Research Department, Naval Postgraduate School
*      Procedures called from Procs and Funcs.
*      Prints hypothetical crises onto screen. This module should
*      be modified to reflect user applicatoins
*****/

```

```

void   PrintCodes()
/*
    Called from procedure DetermineCrisis. Prints various
    crisis options onto console window.
*/
{
    printf( "[ 1 ] \t Coup d'Etat \n" );
    printf( "[ 2 ] \t Bombing of US/Allied Embassy or Base \n" );
    printf( "[ 3 ] \t Revolution in country of interest \n" );
    printf( "[ 4 ] \t Civil War in country of interest \n" );
    printf( "[ 5 ] \t Territorial Dispute of interest \n" );
    printf( "[ 6 ] \t Hostage Crisis \n" );
    printf( "[ 7 ] \t Attack on US forces \n" );
    printf( "[ 8 ] \t Humanitarian Aid \n" );
    printf( "[ 9 ] \t Non-combatant Evacuation \n" );
    printf( "[ 10 ] \t UN Maritime Interdiction \n" );
}

```

```

/*****
*   Presence Calculation Module
*   Written by LT Takashi R. Yamamoto
*   Operations Research Department, Naval Postgraduate School
*
*   Called from main to calculate current ship presence
*
*****/

#include <stdio.h>
#include <stdlib.h>
#include "structs.h"

extern struct Posit    CVLocation[];
extern struct ship     *CV[];
extern short    crisis, region;

struct    Posit    *LastPtr, *currentPtr;

void    BuildArray( struct Posit *currentPtr, short cv, short p, short k )
    /*
        Called from procedure ReadLocation. Creates a matrix
        of surface ship location, based on the data in the
        external text file "location."

        */
{
    (*currentPtr).period = p;
    (*currentPtr).location = k;
    (*currentPtr).next = NULL;

    if ( CVLocation[ cv ].period == 0 )
    {
        CVLocation[ cv ] = *currentPtr;
        LastPtr = currentPtr;
    }
    else
    {
        ( *LastPtr ).next = currentPtr;
        LastPtr = currentPtr;
    };    /* end 'if' loop */
}    /* end procedure BuildArray */

void    FlushFile( FILE *fp )
    /*

```

Called by various procedures. Flushes data stream.

```
        */
{
    short  c;

    while( ( c = fgetc( fp ) ) != '\n' ) && ( c != EOF ) )
        ;

}    /* end procedure FlushFile */
```

```
void  UpdateStatus()
    /*
```

Called from procedure ReadLocation. Opens the external files containing status and make-ready data for each ship. Reads the appropriate data into the corresponding fields of the ship type data structure.

```
        */
{

    FILE  *fp, *fp2;
    short  state,cv, mrt;

    fp = fopen( "Status", "r" );
    fp2 = fopen( "Make-ready", "r" );

    if ( fp == NULL )
    {
        printf( "The ship status file cannot be found. \n" );
        printf( "Please ensure that the file 'Status' is in the current directory." );
        exit(0);
    };

    if ( fp2 == NULL )
    {
        printf( "The ship status or Make-ready file cannot be found. \n" );
        printf( "Please ensure that the file 'Make-ready' is in the current directory." );
        exit(0);
    };

    for( cv = 0; cv <= numCV - 1; cv++ )
    {
        fscanf( fp, "%d", &state );
        FlushFile( fp );
        fscanf( fp2, "%d", &mrt );
        FlushFile( fp2 );
        CV[ cv ] = malloc( sizeof( struct ship ) );
        ( *CV[ cv ] ).ID = cv + 1;
        ( *CV[ cv ] ).status = state;
    }
}
```

```

        ( *CV[ cv ] ).days = mrt;
    };

    fclose( fp );
    fclose( fp2 );
}

/***** read in Location array *****/

void ReadLocation()
    /*
        Called from the main program. This procedure opens a text
        file containing the locations of ships within the area of
        interest. The completed matrix will have the presence of
        ships the current period, and for the next eleven periods
        ( approximately one quarter ) where each period is two
        weeks in length.

        */
{
    FILE *fp;
    short k,cv,p;
    char c;

    fp = fopen( "Location", "r" );

    if ( fp == NULL )
    {
        printf( "The ship location file cannot be found. Please ensure \n" );
        printf( "that the file 'Location' is in the current directory." );
        exit(0);
    };

    p = 0;
    cv = 0;
    while ( ( c = fgetc(fp)) != EOF )
    {
        while (( c != '\n' ) && ( c != EOF ))
        {
            if (( c != '\t' ) && ( c != ' ' ))
            {
                p++;
                k = ChrToDigit( c );
                currentPtr = malloc( sizeof( struct Posit ) );
                BuildArray( currentPtr, cv, p, k );
            };
            c = fgetc(fp);
        }
    }
}

```

```
        cv++;  
        p = 0;  
    };  
    fclose( fp );  
  
    UpdateStatus();  
  
}
```

```

/*****
*      FDO Determination Module
*      Written by LT Takashi R. Yamamoto
*      Operations Research Department, Naval Postgraduate School
*
*      Based on user entered crisis code, determines the
*      generic FDO package required to meet contingency.
*****/

```

```

#include <stdio.h>
#include <stdlib.h>
#include "Globals.c"

```

```

short  FDORequired( short criCode )
/*

```

```

    This function will return the FDO code, based on the crisis
    code that is passed to the function. The FDOs used in this
    program are HYPOTHETICAL ONLY and do not reflect the actual
    codes utilized by CINCPACFLT. This is insure that the program
    remains unclassified. For realism, programmers should alter
    the function as required.
    */

```

```

{

```

```

    short  level, code;

```

```

    switch( criCode )
    {

```

```

        case 5:

```

```

        /*      Territorial Dispute with one friendly involved.
        Obviously, the response to this situation would
        vary with the level of force deemed necessary and
        the nations involved. For the purposes of this
        project, it will require 2 CVBGs. */

```

```

        code = 21;          /* 2 CVBGs, 1 ARGs */
        break;

```

```

        case 6:

```

```

        /*      US citizens in foreign nation taken as hostages by host nation
        Initially, this situation would require a 2 CVBG and possibly
        and ARG, however, for the purposes of this project, only the
        CVBG force will be deployed. */

```

```

        code = 21;          /* 2 CVBGs, 1 ARG      */
        break;

```

```

        case 7:

```

```

        /*      Military attacks on US forces overseas */

```

```

        code = 32;          /* 3 CVBGs, 2 ARGs      */
        break;

```

```

case 8:
    /*      Natural Disaster Relief/Humanitarian Aide */

    printf( "What level of relief is required?: " );
    scanf( "%d", &level );
    switch( level )
    {
        case 1:
            /* Level One - the lowest */

            code = 11;
            break;

        case 2:
            /* Level Two - mid-level */

            code = 12;
            break;

        case 3:
            /* Level Three - catastrophic */

            code = 13;
            break;

        default:

            code = 1;
    };
    break;

case 9:
    /* Non-combatant Evacuation */

    code = 11;
    break;

case 10:
    /*      Enforcing UN trade sanctions on SLOCs */

    code = 30;
    break;

default:
    /*      The default value represents a crisis that requires only a
        a single CVBG. */

    code = 10;
};

return( code );
}

```

```

/*****
*   Procedures and Functions Module
*   Written by LT Takashi R. Yamamoto
*   Operations Research Department, Naval Postgraduate School
*
*   General procedures and functions used by ECRM.
*   Except some functions for Dial's Algorithm, which remain
*   in the Dial module
*****/

#include <stdio.h>
#include <stdlib.h>
#include "Globals.c"

short  portStops[ maxNodes ][ maxNodes ];
struct  ship  *shipQ, *pQIndex;

/*****/

void  Initialize()
{
    short  i;

    for( i = 0; i <= maxNodes; i++ )
        adjList[ i ].pAdjNodes = NULL;
    for( i = 0; i <= maxEdge; i++ )
        B[ i ].pTr = NULL;
    setS = NULL;
    good = false;
    xsitSpeed = speed;
}

    /*** end of Initialize ***/

short  ChrToDigit( short khr )
    /*
        Converts character types to integer types.
        */
{
    return( khr - '0' );
}

void  DetermineCrisis()
    /*
        Although this program requires the user to manually enter
        the crisis code and region code, the final program should
        also be able to obtain this data from CASES directly.
        */
{
    printf( "US Navy Ship Employment and Crisis Response Model \n\n\n\n" );
    PrintCodes();
}

```

```

while( !good )
{
    printf( "\nEnter crisis category. " );
    scanf( "%d", &crisis );
    good = ( crisis >= 0 ) && ( crisis < 11 );
    if( !good )
        printf( "Invalid entry. Try again. \n" );
};
printf( "\n" );
good = false;
fDO = FDORequired( crisis );

while( !good )
{
    printf( "Enter crisis region: " );
    scanf( "%d", &source );
    good = ( source > 0 ) && ( source <= maxNodes );
    if( !good )
        printf( "Invalid entry. Try again. \n" );
};
printf( "\n" );
}

void DetermineStops()
/*
    Called from procedure DetermineDistance. Inputs the
    port stops between different nodes and stores the
    data in a two-dimensional array.
*/
{
    FILE *fp;
    char c;
    short k, x = 0, y = 0;

    fp = fopen( "Stops", "r" );
    if ( fp == NULL )
    {
        printf( "The port stop file cannot be found. Please ensure \n" );
        printf( "that the file 'Stops' is in the current directory." );
        exit(0);
    };

    while ( ( c = fgetc(fp)) != EOF )
    {
        while (( c != '\n' ) && ( c != EOF ))
        {
            if (( c != '\t' ) && ( c != ' ' ))
            {
                k = ChrToDigit( c );
            }

```

```

        portStops[ x ][ y ] = k;
        y++;
    };
    c = fgetc(fp);
}
x++;
y = 0;
};
fclose( fp );
}

void DetermineDistance()
/*
    Called from main. Updates ship fields, based on the
    distances calculated by Dial. Updates the location,
    status, days and stops fields for each ship.
*/
{
    struct nodeSet *pDistanceQ;
    short c, posit, steamingDays;
    Boolean found;

    DetermineStops();
    for( c = 0; c <= numCV - 1; c++ )
    {
        pDistanceQ = setS;
        found = false;
        posit = CVLocation[ c ].location;
        while( !found ) /* no cv in region */
        {
            if( pDistanceQ == NULL )
            {
                printf( "CV not in this theater. \n" );
                ( *CV[ c ] ).status = 4;
                ( *CV[ c ] ).days = 30;
                found = true;
            } /* If unable to locate CV in theater */
            else
            {
                found = ( ( *pDistanceQ ).node == posit );
                if( found )
                {
                    ( *CV[ c ] ).location = posit;
                    ( *CV[ c ] ).distance = ( *pDistanceQ ).distance;
                    steamingDays = ( ( ( *CV[ c ] ).distance ) / xsitSpeed ) /
24;

                    if( ( ( *CV[ c ] ).status == 4 ) && ( posit != source ) )
                        ( *CV[ c ] ).days = 30;
                    else

```

```

steamingDays;

(*CV[ c ] ).days = ( *CV[ c ] ).days +

(*CV[ c ] ).stops = portStops[ source - 1 ][ posit - 1 ];
} /* end 'if found' */
else
    pDistanceQ = ( *pDistanceQ ).pNext;
};
};
(*CV[ c ] ).days = ( *CV[ c ] ).days + ( *CV[ c ] ).stops;
(*CV[ c ] ).cP = 1;
(*CV[ c ] ).hV = 1;

}; /* end CV loop */
} /* end of procedure DetermineDistance */

```

/****** Procedures and Functions for Heap Sort *****/

```

short Left( short i )
/*
    Called from procedure heapify. Determines index
    for left node.
*/
{
    return( 2 * i );
} /* end of procedure Left */

```

```

short Right( short i )
/*
    Called from procedure heapify. Determines index
    for right node.
*/
{
    return( ( 2 * i ) + 1 );
} /* end of procedure Right */

```

```

short Exchange( short i, short largest )
/*
    Called from procedure Heapify. Exchanges the
    nodes in the array.
*/
{
    struct ship *temp;

```

```

    temp = CV[ i ];
    CV[ i ] = CV[ largest ];
    CV[ largest ] = temp;
)    /** end of procedure Exchange ***/

void  Heapify( short i, short heapSize )
    /*
        Called from procedure SortQueue, and recursively
        to establish heap array.
    */
{
    short  l, r, largest;

    l = Left( i );
    r = Right( i );
    if( ( l <= heapSize ) && ( ( *CV[ l ] ).status > ( *CV[ i ] ).status ) )
        largest = l;
    else
        largest = i;

    if( ( r <= heapSize ) && ( ( *CV[ r ] ).status > ( *CV[ largest ] ).status ) )
        largest = r;

    if( largest != i )
    {
        Exchange( i, largest );
        Heapify( largest, heapSize );
    };
}    /** end of procedure Heapify ***/

void  SortQueue()
    /*
        Called from main. Organizes Dial solution into heap array, then
        sorts the array by status first, using a heap sort, then by
        distance by using linear comparisons.
    */
{
    short  c, i, heapSize;

    heapSize = numCV;
    c = heapSize/2;
    while( c >= 1 )
    {
        Heapify( c - 1, heapSize - 1 );
        c--;
    };

    c = heapSize;

```

```

while( c >= 2 )
{
    Exchange( 0, c - 1 );
    heapSize--;
    Heapify( 0, heapSize - 1 );
    c--;
};

for( c = 0; c <= numCV - 2; c++ )
    if( ( *CV[ c ] ).status == ( *CV[ c + 1 ] ).status )
        if( ( *CV[ c ] ).distance > ( *CV[ c + 1 ] ).distance )
        {
            Exchange( c, c + 1 );
        }
        else
        {
            if( ( *CV[ c ] ).distance == ( *CV[ c + 1 ] ).distance )
                Exchange( c, c + 1 );
        }
};
shipQ = CV[ 0 ];
for( c = 0; c <= numCV - 2; c++ )
{
    ( *CV[ c ] ).pNext = CV[ c + 1 ];
    ( *CV[ c + 1 ] ).pBack = CV[ c ];
};
( *CV[ 0 ] ).pBack = NULL;
}    /** end procedure SortQueue ***/

```

/****** Procedures and Functions for display *****/

```

void    ChangeQueue()
    /*
        Called to change the order of the ship queue.

        */
{
    short    ID, pos, c = 0;
    struct    ship    *pTemp;
    Boolean    good = false;

    while( !good )
    {
        printf( "\nWhich ship would you like to change? " );
        scanf( "%d", &ID );
        if( ( ID < 0 ) || ( ID > numCV ) )
            printf( "Not a valid entry. Please re-enter. \n" );
    }
}

```

```

        else
            good = true;
    };

    good = false;
    while( !good )
    {
        printf( "\nWhere should I put it in the queue? " );
        scanf( "%d", &pos );
        if( ( pos < 0 ) || ( pos > numCV ) )
            printf( "Not a valid entry. Please re-enter. \n" );
        else
            good = true;
    };

    while( ( *CV[ c ] ).ID != ID )
        c++;
    ( *CV[ c - 1 ] ).pNext = ( *CV[ c ] ).pNext;
    ( *CV[ c + 1 ] ).pBack = ( *CV[ c ] ).pBack;
    ( *CV[ pos - 2 ] ).pNext = CV[ c ];
    ( *CV[ pos - 1 ] ).pBack = CV[ c ];
    ( *CV[ c ] ).pNext = CV[ pos - 1 ];
    ( *CV[ c ] ).pBack = CV[ pos - 2 ];

    if( pos == 1 )
        shipQ = CV[ c ];
    pTemp = shipQ;
    for( c = 0; c <= numCV - 1; c++ )
    {
        CV[ c ] = pTemp;
        pTemp = ( *pTemp ).pNext;
    };
}

    /** end of procedure ChangeQueue */

void ChangeStatus()
{
    short ID, state, c = 0;
    Boolean good = false;

    while( !good )
    {
        printf( "\nWhich ship would you like to change? " );
        scanf( "%d", &ID );
        if( ( ID < 0 ) || ( ID > numCV ) )
            printf( "Not a valid entry. Please re-enter. \n" );
        else
            good = true;
    };

    good = false;

```

```

while( !good )
{
    printf( "\nWhat is its new status? " );
    scanf( "%d", &state );
    if( ( state < 1 ) || ( state > 5 ) )
        printf( "Not a valid entry. Please re-enter. \n" );
    else
        good = true;
};

while( ( *CV[ c ] ).ID != ID )
    c++;
if( ( *CV[ c ] ).status == 4 )
    ( *CV[ c ] ).days = (( *CV[c] ).distance / 24 / xsitSpeed) + ( *CV[c] ).stops;
( *CV[ c ] ).status = state;
SortQueue();
}    /** end of procedure ChangeStatus ***/

void ChangeStops()
{
    short ID, stops, c = 0;

    Boolean good = false;

    while( !good )
    {
        printf( "\nWhich ship would you like to change? " );
        scanf( "%d", &ID );
        if( ( ID < 0 ) || ( ID > numCV ) )
            printf( "Not a valid entry. Please re-enter. \n" );
        else
            good = true;
    };

    printf( "\nHow many days in port? " );
    scanf( "%d", &stops );
    while( ( *CV[ c ] ).ID != ID )
        c++;
    ( *CV[ c ] ).days = ( *CV[ c ] ).days - ( *CV[ c ] ).stops;
    ( *CV[ c ] ).stops = stops;
    ( *CV[ c ] ).days = ( *CV[ c ] ).days + ( *CV[ c ] ).stops;
}    /** end of procedure ChangeStops ***/

void ChangeMRT()
    /*
        Called from procedure GenerateTable. Allow user
        to change unit make-ready times.
    */

```

```

{
    short  ID, mrt, c = 0, steamingDays;

    Boolean    good = false;

    while( !good )
    {
        printf( "\nWhich ship would you like to change? " );
        scanf( "%d", &ID );
        if( ( ID < 0 ) || ( ID > numCV ) )
            printf( "Not a valid entry. Please re-enter. \n" );
        else
            good = true;
    };

    printf( "\nWhat is the new make-ready time (in days)? " );
    scanf( "%d", &mrt );

    while( ( *CV[ c ] ).ID != ID )
        c++;
    ( *CV[ c ] ).days = mrt;
    steamingDays = ( ( ( *CV[ c ] ).distance ) / xsitSpeed ) / 24;
    if( ( ( *CV[ c ] ).status == 4 ) && ( ( *CV[ c ] ).location != source ) )
        ( *CV[ c ] ).days = 30;
    else
        ( *CV[ c ] ).days = ( *CV[ c ] ).days + steamingDays;
}

    /** end of procedure ChangeMRT **/

void  ChangeSpd()
    /*
        Called from procedure GenerateTable. Allows user
        to change the standard response speed of 18 kts.
    */

{
    short  c, spd, steamingDays;

    printf( "\nWhat is the new transit speed (in knots)? " );
    scanf( "%d", &spd );
    for( c = 0; c <= numCV - 1; c++ )
    {
        steamingDays = ( ( ( *CV[ c ] ).distance ) / xsitSpeed ) / 24;
        ( *CV[ c ] ).days = ( *CV[ c ] ).days - steamingDays;
        steamingDays = ( ( ( *CV[ c ] ).distance ) / spd ) / 24;
        ( *CV[ c ] ).days = ( *CV[ c ] ).days + steamingDays;
    };
}

    /** end of procedure ChangeSpd **/

void  GenerateTable()

```

```

/*
    Called from main. Generates a table representing the
    ship queue in tabular form, in order of preference.
    Allows user to modify parameters as necessary.
*/

```

```

{
    short  c, opt;
    char   chg;
    Boolean valid = false;

    good = false;
    while( !good ) /* while not happy with the table... */
    {
        printf( "CV \t Status \t Location \t Distance \t Port Days \t Days \n" );
        for( c = 0; c <= numCV - 1; c++ )
        {
            printf( "%d \t %d \t\t %d \t\t", (*CV[c]).ID, (*CV[c]).status,
                    (*CV[c]).location );
            printf( " %d \t\t %d \t\t %d \n", (*CV[ c ]).distance,
                    (*CV[ c ]).stops, (*CV[ c ]).days );
        };
        valid = false;
        while( !valid ) /* keyboard input isn't valid! */
        {
            printf( "\nWould you like to make a change? " );
            FlushFile( stdin );
            scanf( "%c", &chg );
            valid = ( (chg == 'Y') || (chg == 'y') || (chg == 'n') || (chg == 'N') );
            if( !valid )
                printf( "\nNot a valid entry. Please try again. \n" );
            if( ( chg == 'Y' ) || ( chg == 'y' ) )
            {
                printf( "\n\t[ 1 ]\tChange Ship Queue\n" );
                printf( "\t[ 2 ]\tChange Ship Status\n" );
                printf( "\t[ 3 ]\tChange Number of Port Stops\n" );
                printf( "\t[ 4 ]\tChange Ship Make-ready Time\n" );
                printf( "\t[ 5 ]\tChange Transit Speed\n" );
                printf( "\t[ 6 ]\tNo Change\n" );
                printf( "\nSelect option: " );
                scanf( "%d", &opt );
                switch( opt )
                {
                    case 1:
                        ChangeQueue();
                        break;

                    case 2:
                        ChangeStatus();
                        break;

                    case 3:
                        ChangeStops();

```

```

                                break;

                                case 4:
                                    ChangeMRT();
                                    break;

                                case 5:
                                    ChangeSpd();
                                    break;

                                default:
                                    good = true;

                                );          /* end of switch case */

                                }          /* end of if change = yes */
                                else
                                    good = true;

                                );          /* end of change loop */

                                };          /* end 'while good' loop */

                                }          /*** end of procedure GenerateTable ***/

/***** Procedures for calculating MOEs *****/

short FindDelay( short cv )
    /*
        Called from procedure CalculateMOE. Determines
        delay, in days, for FDO to arrive in the region.
        This value is returned to CalculateMOE.
    */
{
    short t, delay = 0;

    for( t = 0; t <= cv - 1; t++ )
    {
        if( delay <= ( *CV[ t ] ).days )
            delay = ( *CV[ t ] ).days;
    };
    return( delay );
}          /*** end of procedure FindDelay ***/

void OutputMOE2( short ships, short delay, FILE *fp )
    /*
        Called from procedure CalculateMOEs. Calculates the
        total percentage of the FDO requirements on station
    */

```

each day. This data is output to an external ASCII text file called ECRM Output and can be opened by commercially available software for graphical analysis of the response.

```

                                                                    */
{
    short  d = 0, t;
    float  total = 0;

    while( d <= delay )
    {
        for( t = 0; t <= ships - 1; t++ )
        {
            if( ( *CV[ t ] ).days == d )
                total = total + ( *CV[ t ] ).cP;
        };

        fprintf( fp, "%f\t", total/ships );      /* total cP or hV required */
        d++;
    };      /* while the day counter is less than the total delay for the
              FDO to arrive. */

    fprintf( fp, "\n\n\n" );
}

    /*** end of procedure OutputMOE2 ***/

```

```

void  CalculateMOEs()
    /*
        Called from main. Calculates the total number of days
        required for the total FDO package to arrive. This
        information is displayed on screen. The total percentage
        of the FDO package on station each day is output to
        an external file. Also provides data on the FDO package
        one level below, and one level above the recommended
        levels.
                                                                    */
{
    short  delay, cv, cvLow, cvHigh, d, m;
    FILE   *fileOut;

    fileOut = fopen( "ECRM Output", "w" );

    cv = div( fDO, 10 ).quot;
    if( cv == 1 )
        cvLow = cv;
    else
        cvLow = cv - 1;
    cvHigh = cv + 1;

```

```

printf( "\nThis crisis requires %d carrier", cv );
if( cv == 1 )
    printf( ".\n" );
else
    printf( "s.\n" );
printf( "Delay until complete FDO arrives is %d days. \n", delay = FindDelay( cv ) );
OutputMOE2( cv, delay, fileOut );

printf( "For %d carrier", cvLow );
if( cvLow == 1 )
    printf( ", " );
else
    printf( "s, " );
printf( "the delay is %d days.\n", delay = FindDelay( cvLow ) );
OutputMOE2( cvLow, delay, fileOut );

printf( "For %d carriers, the delay is %d days.\n", cvHigh, delay = FindDelay(
cvHigh ) );
OutputMOE2( cvHigh, delay, fileOut );

fclose( fileOut );

}    /** end of procedure CalculateMOEs ***/

```

```

/*****
*   Dial's Algorithm Module
*   Written by LT Takashi R. Yamamoto
*   Operations Research Department, Naval Postgraduate School
*
*   Called from main to implement the shortest-path
*
*****/

#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Globals.c"

/***** PROCEDURES AND FUNCTIONS *****/

void BuildAdjList( struct edge *pNew )
/*
    Called from procedure ReadNetworkData to build
    and update a sparse matrix composed of nodes
    representing adjacent geographic regions. The
    adjacent nodes are assumed to be normal
    transit legs taken by ships.
*/
{
    if( adjList[ ( *pNew ).nodeOut - 1 ].pAdjNodes == NULL )
        adjList[ ( *pNew ).nodeOut - 1 ].pAdjNodes = pNew;
    else
    {
        ( *pNew ).pNext = adjList[ ( *pNew ).nodeOut - 1 ].pAdjNodes;
        adjList[ ( *pNew ).nodeOut - 1 ].pAdjNodes = pNew;
    };
}

/**** end BuildAdjList ****/

void MakeNodeSet( struct nodeSet *pVert )
/*
    Called from procedure ReadNetworkData to build
    a linked list of nodes representing geographic
    regions in the Pacific theater.
*/
{
    if( pFirst == NULL ) /* if it's the first node */
    {
        pFirst = pVert;
        pLast = pFirst;
    }
}

```

```

else                                     /* all other nodes that follow */
{
    ( *pLast ).pNext = pVert;
    (*pVert).pBack = pLast;
    pLast = ( *pLast ).pNext;
};
}    /* end MakeNodeSet */

void CreateBucket( short source )
/*
    Called from procedure ReadNetworkData to build
    an array of "buckets" where the index represents
    the distance from the source node to other regions.

*/
{
    struct nodeSet    *pIndex, *pIndex2;

    pIndex = pFirst;

    if( ( *pIndex ).node == source ) /* if first node is source node */
    {
        B[ maxEdge ].pTr = ( *pIndex ).pNext;
        ( *( *pIndex ).pNext ).pBack = NULL;
    }
    else                                     /* source node is not first node */
    {
        B[ maxEdge ].pTr = pIndex;
        while( ( *pIndex ).node != source )
            pIndex = ( *pIndex ).pNext;
        ( *( *pIndex ).pBack ).pNext = ( *pIndex ).pNext;
        ( *( *pIndex ).pNext ).pBack = ( *pIndex ).pBack;
        pIndex2 = ( *pIndex ).pBack;
    };

    ( *pIndex ).pNext = NULL;
    ( *pIndex ).pBack = NULL;
    B[ 0 ].pTr = pIndex;                    /* places source region into */
    ( *B[ 0 ].pTr ).distance = 0;           /* first location.                */
    ( *B[ 0 ].pTr ).bucket = 0;
    if( ( *pIndex2 ).pNext == NULL )
        pLast = pIndex2;

}    /* end CreateBucket */

void ReadNetworkData( short source )
{
    /*

```

Called from procedure Dial to read in the graph data representing regions in the Pacific, and the DMA distances associated with traveling between those points.

*/

```

short  nodeOut, nodeIn, i, dist;
struct edge  *pTemp;

while( !(feof( data ) ) )
{
    fscanf( data, "%d %d %d", &nodeOut, &nodeIn, &dist );
    pTemp = malloc( sizeof( struct edge ) );
    (*pTemp).nodeOut = nodeOut;
    (*pTemp).nodeIn = nodeIn;
    (*pTemp).distance = dist;
    (*pTemp).pNext = NULL;
    BuildAdjList( pTemp );
    pTemp = malloc( sizeof( struct edge ) );
    (*pTemp).nodeOut = nodeIn;
    (*pTemp).nodeIn = nodeOut;
    (*pTemp).distance = dist;
    (*pTemp).pNext = NULL;
    BuildAdjList( pTemp );
};

for( i = 1; i <= maxNodes; i++ )
{
    pVertex = malloc( sizeof( struct nodeSet ) );
    ( *pVertex ).node = i;
    ( *pVertex ).distance = infinity;
    ( *pVertex ).pNext = NULL;
    ( *pVertex ).pBack = NULL;
    ( *pVertex ).bucket = maxEdge;
    P[ i ] = pVertex;
    MakeNodeSet( pVertex );
};

CreateBucket( source );
fclose( data );
}

/** end of ReadData */

/** procs and funcs for Dial's */

struct  nodeSet      *GetNode( short i )
/*
    Called from procedure FindMIn to "pull" the
    next node from the distance bucket. This node

```

represents the next closest region.

```

    */
{
    struct nodeSet    *pTemp;

    pTemp = B[ i ].pTr;
    B[ i ].pTr = ( *B[ i ].pTr ).pNext;
    return( pTemp );
}    /** end of GetNode ***/

```

```

void    Update( struct nodeSet *V )
    /*

```

Called from procedure FindMin to update the distance bucket and pointer array after getting the next node.

```

    */
{
    struct edge    *J;
    struct nodeSet    *marker;
    short    mod;

    J = adjList[ (*V).node - 1 ].pAdjNodes;
    while( J != NULL )
    {
        if( ( (*P[ (*J).nodeIn ]).distance ) >
            ( (*V).distance + (*J).distance ) )
        {
            /*If distance in P is greater than V.distance + J.distance */
            (*P[ (*J).nodeIn ]).distance = ( (*V).distance + (*J).distance );
            mod = fmod( (*P[ (*J).nodeIn ]).distance, maxEdge );
            marker = B[ mod ].pTr;

            if( B[ mod ].pTr == NULL )                /* Bucket is empty */
            {
                if( (*P[ (*J).nodeIn ]).pBack == NULL )
                {
                    B[ (*P[ (*J).nodeIn ]).bucket ].pTr
                        = (*P[ (*J).nodeIn ]).pNext;
                    ( (*P[ (*J).nodeIn ]).pNext ).pBack = NULL;
                    (*P[ (*J).nodeIn ]).pNext = NULL;
                    B[ mod ].pTr = &(*P[ (*J).nodeIn ]);
                    (*P[ (*J).nodeIn ]).bucket = mod;
                }
                else    /* not first node in line */
                {
                    ( (*P[ (*J).nodeIn ]).pBack ).pNext

```

```

        = (*P[ (*J).nodeIn ]).pNext;
        ( (*P[ (*J).nodeIn ]).pNext ).pBack
        = (*P[ (*J).nodeIn ]).pBack;
        (*P[ (*J).nodeIn ]).pNext = NULL;
        (*P[ (*J).nodeIn ]).pBack = NULL;
        B[ mod ].pTr = &(*P[ (*J).nodeIn ]);
        (*P[ (*J).nodeIn ]).bucket = mod;
    };

}
else /* end of "if bucket empty" */
/* Bucket not empty */
{
    while( (*marker ).pNext != NULL )
        marker = ( *marker ).pNext;

    if( (*P[ (*J).nodeIn ]).pBack == NULL )
    {
        B[ (*P[ (*J).nodeIn ]).bucket ].pTr
            = (*P[ (*J).nodeIn ]).pNext;
        ( (*P[ (*J).nodeIn ]).pNext ).pBack = NULL;
        (*P[ (*J).nodeIn ]).pNext = NULL;
        ( *marker ).pNext = &(*P[ (*J).nodeIn ]);
        (*P[ (*J).nodeIn ]).pBack = marker;
        (*P[ (*J).nodeIn ]).bucket = mod;
    }
    else /* not first node in line */
    {
        ( (*P[ (*J).nodeIn ]).pBack ).pNext
            = (*P[ (*J).nodeIn ]).pNext;
        ( (*P[ (*J).nodeIn ]).pNext ).pBack
            = (*P[ (*J).nodeIn ]).pBack;
        (*P[ (*J).nodeIn ]).pNext = NULL;
        (*P[ (*J).nodeIn ]).pBack = NULL;
        ( *marker ).pNext = &(*P[ (*J).nodeIn ]);
        (*P[ (*J).nodeIn ]).pBack = marker;
        (*P[ (*J).nodeIn ]).bucket = mod;
    }
};

/* end of "if bucket is empty" */

}; /* end of "if P[d] > V[d] + J[d]" */
J = ( *J ).pNext;

}; /* end of "while" loop */

} /* *** end Update *** /

```

```

void FindMin()
/*

```

Called from procedure Dial. Calls function GetNode
to find next nearest node. Then updates the set S

which is a linked list of the nodes in order of distance from the source. This list is also stored in the pointer array P. Lastly, calls the procedure Update to update the distance bucket and pointer array after obtaining a node.

```

                                                                    */
{
    short  x;
    struct  nodeSet      *Vertex;

    while( B[ ( x = fmod( D, maxEdge ) ) ].pTr == NULL )
        D++;          /* looking for a bucket that isn't empty */
    Vertex = GetNode( x );
    if( setS == NULL )
    {
                                                                    /* updates set S */
        setS = Vertex;
        sLast = setS;
    }
    else
    {
        ( *sLast ).pNext = Vertex;
        sLast = ( *sLast ).pNext;
    };
    Update( Vertex );
}    /*** end of FindMin ***/

void  Dial()
    /*
        Called from ECRM main to implement Dial's
        Algorithm, a modification of the Dijkstra
        algorithm, to obtain the shortest-path
        distances from a source node to all nodes
        in the network.
                                                                    */
{
    short  count;

    data = fopen( "Network", "r" );
    ReadNetworkData( source );
    D = 0;
    count = 0;
    while( count != maxNodes )
    {
        count++;
        FindMin();
    };
}    /*** end Dial ***/

```

```

/*****
*      USN Surface Ship Employment and Crisis Response Model      *
*
*      Written by LT Takashi R. Yamamoto
*      Operations Research Department, Naval Postgraduate School *
*      March, 1993
*****/

```

```

#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include "structs.h"

```

```

/***** GLOBAL VARIABLES *****/

```

```

FILE      *data;
short     crisis, region, source, D, i, xsitSpeed;
short     fDO;
Boolean    good;
struct    node    adjList[ maxNodes ];
struct    pNode   B[ maxEdge + 1 ];
struct    nodeSet   *P[ maxNodes + 1 ];
struct    nodeSet   *pFirst, *pLast, *setS, *sLast, *sMarker;
struct    nodeSet   *pVertex;
struct    Posit    CVLocation[ numCV ];
struct    ship     *CV[ numCV ];

```

```

/***** MAIN CODE *****/

```

```

main()
{
    Initialize();

    ReadLocation();          /* Reads in CV location matrix and stores
                             the data in a linked list.      */

    DetermineCrisis();       /* Gets crisis code and region code
                             Currently limited to a single crisis
                             and region.                      */

    Dial();                 /* Calls Dial's Algorithm to compute the
                             shortest-path distances from the source
                             region to other regions in the theater. */

    DetermineDistance();     /* Establishes distance queue, updates ship
                             type data structures for each ship.    */

    SortQueue();            /* Creates ship queue, then uses heap sort to
                             sort queue by status and distance.     */
}

```

```
GenerateTable();          /* Generates output of queue options on
                           console screen. User can modify response
                           queue as necessary. */

CalculateMOEs();          /* Calculates measures of effectiveness. */

}
```

APPENDIX B

GEOGRAPHIC LOCATION CODES

- | | |
|---|---|
| 1 | Eastern Pacific (San Francisco, San Diego, Seattle) |
| 2 | Mid-Pacific (Pearl Harbor) |
| 3 | Northwestern Pacific (Yokosuka) |
| 4 | Southwestern Pacific (Philippines) |
| 5 | Southwestern Pacific (Singapore) |
| 6 | Indian Ocean |

SHIP STATUS CODES

- | | |
|---|--------------------------------|
| 1 | In Transit / Local Operations |
| 2 | Upkeep |
| 3 | Returning from Deployment |
| 4 | Not Available (Employed) |
| 5 | Major Maintenance Availability |

APPENDIX C

HYPOTHETICAL FDO CRISIS CODES

Crisis Code	Title	Description	FDO Code
1	Coup d'Etat	Coup in "friendly" nation	10
2	Bombing	Bombing of embassy/base	10
3	Revolution	Revolution in country of interest	10
4	Civil War	Civil War in country of interest	10
5	Territorial Dispute	Friendly nation involved LRC/LIC	20
6	Hostages	US/Allied hostages	21
7	Attack on US Forces	MRC	32
8	Humanitarian Aid	Non-disaster	various
9	Non-combatant Evacuation	Rescue of citizens	11
10	Blockade	UN Maritime Interdiction	30

REFERENCES

- [1] Center for Naval Analyses Research Memorandum 91-79, *A Method for Estimating CVBG Presence and Crisis Response*. by Kevin J. Becker, October 1991.
- [2] Center for Naval Analyses Research Working Paper 1640.09, *The Arithmetic of Naval Peacetime Presence and Crisis Response*. by John V. Hall, 6 August 1990.
- [3] Naval Surface Warfare Center NSWCDD/TR-92/250, *Peacetime Deployment Spreadsheet*. by Michael J. Lindemann, 1 June 1992.
- [4] Naval War College. "Game Book." Naval Surface Warfare 2030 Symposium II, 3 December 1991.
- [5] Bush, George H. *National Security Strategy of the United States*. The White House: Washington, D.C., August 1991.
- [6] DeltaPoint Inc., *DeltaGraph Professional*, Monterey, 1991.

BIBLIOGRAPHY

- Anderson, Bruce and John Flynn. "CASES: A System for Assessing Naval Warfare Capability." Joint Directors of Labs C3I Symposium, 1990.
- Bolt, Beranek and Newman. "CASES." Models Review Board, San Diego, 16 June 1992.
- Bush, George H. *National Security Strategy of the United States*. The White House: Washington, D.C., August 1991.
- Center for Naval Analyses Research Memorandum 91-79, *A Method for Estimating CVBG Presence and Crisis Response*. by Kevin J. Becker, October 1991.
- Center for Naval Analyses Research Working Paper 1640.09, *The Arithmetic of Naval Peacetime Presence and Crisis Response*. by John V. Hall, 6 August 1990.
- CINCPACFLT Memorandum 11-92. *Analysis of Naval Presence and Crisis Response*. by Takashi R. Yamamoto, June 1992.
- Corman, Thomas H., Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*. Cambridge: The MIT Press, 1990.
- Naval Surface Warfare Center NSWCDD/TR-92/250, *Peacetime Deployment Spreadsheet*. by Michael J. Lindemann, 1 June 1992.
- Naval War College. "Game Book." Naval Surface Warfare 2030 Symposium II, 3 December 1991.
- O'Keefe, Sean, Kelso, Frank B., and Mundy, Carl E. ". . . From the Sea." *Proceedings*, November, 1992. pp. 93-96.
- Runyan, Ray. "FPC Models Board." Models Review Board, San Diego, 16 June 1992.
- Weeks, Stan. "Crafting a New Maritime Strategy." *Proceedings*. Jan 1992. pp. 30-37.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5100	2
3. Prof. W. P. Hughes Dept. of Operations Research, Code 30 Naval Postgraduate School Monterey, CA 93943-5002	1
4. Prof. R. Dell Dept. of Operations Research, Code 30 Naval Postgraduate School Monterey, CA 93943-5002	1
5. Capt. G. Conner Dept. of Operations Research, Code 30 Naval Postgraduate School Monterey, CA 93943-5002	1
6. Dr. R. Runyan Staff CINCPACFLT Code N3C/N5C Box 6 Pearl Harbor, HI 96860-7000	2
7. Chief of Naval Operations N731 Washington, D.C. 20505	1
8. Mr. John Flynn BBN 1300 North 17th St. Arlington, VA 22209	1

9. Lt. Takashi R. Yamamoto
14677 Merced St.
San Leandro, CA 94579

1