Automatic Generation of Best Emergency Routes and Procedures on a Brazilian Frigate

David L. L. Sicuro
Grupo de Sistemas Digitais
Instituto de Pesquisas da Marinha
Rua Ipiru 2 - Ilha do Governador
Rio de Janeiro, RJ 20931-000 BRAZIL
phone: (55) 21 396-2004
sicuro@ipqm.mar.mil.br

Neil C. Rowe[*]
Department of Computer Science
U.S. Naval Postgraduate School
Monterey, CA 93943 USA
phone: (831) 656-2462
Fax: (831) 656-2814
rowe@cs.nps.navy.mil

Abstract

We describe prototype software we have written that addresses the serious problem of management of damage control on naval vessels. Human reasoning and judgment may be strongly affected by stress and panic when several simultaneous events occur in a damaged-ship environment. Thus it would be helpful in times of emergency to dispassionately calculate the best courses of action to follow and suggest them to personnel.

We describe two tested modules for the decision-aid system of the new Niteroi-Class Frigate Damage Control System project: an emergency-route generator for people and casualties, and a smoke-extraction-procedure generator. Our modules gather and organize sensor data and reports about rooms, gateways and equipment. They reason automatically to determine the best routes and procedures, and can present the damage-related information in a concise manner to users both in the command center and throughout the ship.

This system is implemented as an expert system using the CLIPS shell. The ship is represented as a graph where the compartments are nodes and the doors and hatches are the edges between them; costs reflect the difficulty in passing from a compartment to another. For finding emergency routes for people, a cost-minimizing search is done using the current ship data. A different search is done for smoke extraction to determine routes for ventilation of the smoke, with processing that checks conditions and fills in details.

[*] This work was sponsored by the Brazilian Naval Commission in Washington, DC, USA.

1. Introduction

Damage control is a serious issue on naval vessels because of the frequent remoteness of the vessel from support facilities, the presence of highly flammable materials, and the consequences of combat [U.S. Navy, 1999]. Damage control thus requires careful planning and continual training, even more so than with civilian situations [FEMA, 1999; New York Times, 2000].

The Brazilian Navy is extensively modernizing its Niteroi class (Mk 10 type) frigates. These are vessels of around 3700 tons displacement with a crew of around 200. Upgrades are being done on a wide variety of ship systems and include increased capabilities to monitor ship operations and damage control. This provides an opportunity to try out new methods and procedures, including ideas that the U.S. Navy has not tried. The ships are small enough as to make good laboratories for experiments, while not too large as to make experiments unwieldy.

This paper describes prototype software that is part of the new Niteroi-Class Frigate Damage Control System project named CAVFRAG. This project is being developed in the Brazilian Navy Research Institute. The CAVFRAG project attempts to update the current damage-control system and simplify damage-control procedures. Its objectives are to make data acquisition more reliable and complete; make the system-operator interaction easier and more user-friendly through the effective use of computer graphics; provide damage-related information to the command center and throughout the ship; and provide intelligent guidance to the operator in emergency situations. The work reported here addresses the last requirement.

2. Problem Definition

We first investigated how a decision-aid system could be useful in a damage-control system. After several interviews with the damage-control operators, we concluded that they had very good skills and training and knew exactly what to do during a damage event. However, under stress and panic conditions, when several damage events are happening at the same time, how much is their reasoning and judgement affected? It is difficult to say, but it may well be hard for them to tell which ship sectors would be safe or use all remediation tools with maximum effectiveness.

Our first idea was to build a decision aid to help the operator in tasks directly related to life-saving. Based on statistics, smoke is the main cause of death on naval vessels. So two modules were developed: a route generator for people and casualties, that finds the best safe routes for moving inside the ship, and a smoke extraction procedures generator, that finds the best extraction routes, procedures, and equipment.

3. Decision Aid System Design

If a decision-aid system is to assume some of the job of a person, it should have tranquility, coolness, quick reasoning, and a deep knowledge of the ship plan, so that it may be a dispassionate and quick expert. We decided to implement such a decision-aid system as an expert system, an artificial-intelligence module, inside of CAVFRAG.

Expert systems are defined as any software that embodies non-trivial expertise, knowledge that is known by only a few people in the world, and can reason about it [Gonzalez and Dankel, 1993]. Building an expert system is “knowledge engineering”. Expert systems can use many kinds of artificial-intelligence ideas -- rule-based systems, search, blackboards, relaxation, resolution, etc. -- but rule-based systems are the most popular. The main categories of expert-system applications are diagnosis, inferring causes of observations; classification, inferring categories that apply to observations; control, monitoring and acting in real time; problem-solving, heuristic searching to find an answer; design, and construction of a multi-part solution to a problem [Rowe, 1988].

Rule-based systems have a non-procedural interpretation. This means there is no pre-established program sequence to be followed as in procedural languages like C, Pascal, or Java. Rule-based systems comprise rules, facts and an inference engine. Facts are the “truths” of the system at a certain moment of the program execution; rules are structures that combine facts to produce an action. A rule has the structure of “if <condition> then <action>” and the action can produce new facts that can be used in other rules. The exact order in which rules are tried depends on the particular inference engine used. The expert’s knowledge is embedded in these facts and rules, called collectively a “knowledge base”.

Expert-system shells provide assistance in developing rule-based systems. The shell used in this project is a forward-chaining one called CLIPS [Giarratano, 1998]. The C Language Integrated Production System (CLIPS) is an expert-systems tool developed by Software Technology Branch (STB) of the U.S. NASA (National Aeronautics and Space Administration) organization at the Lyndon B. Johnson Space Center. Since its first release in 1986, CLIPS has undergone continual refinement and improvement. It is now used by thousands of people around the world. CLIPS is designed to facilitate the development of software to model human knowledge or expertise.

4. Knowledge Base

For expert systems, it is important to define the structure of the knowledge base.

- In the route generator for people and casualties, knowledge concerns the structural plan of the ship, the difficulty of using gateways such as doors, hatches, and stairs, how difficulty is affected by the presence of casualties, and the difficulty of traversing damaged (e.g. flooded or smokey) rooms. Sensor data acquired over the ship is also included in the knowledge base.

- In the smoke-extraction procedure generator, knowledge concerns a different aspect of the structural plan of the ship. Here we consider the ship as a set of sealed sectors, each including many compartments. Expertise concerns the paths by which smoke be safely removed by flowing through a sequence of sectors, and how to optimize the use of ventilation and extraction devices. Sensor data is also used.

The frigates of the Niteroi class have almost identical structural plans. This plan is represented in the knowledge base by a graph. Each node of this graph is a representative point in threedimensional space (x, y, and z) for a ship compartment. The representative points are approximately the centers of compartments, and were obtained from measurements on a blueprint of the ship plan. An undirected edge connects two nodes of the graph if at least one real “gateway”(passage, stairway, door, normal hatch, or scuttle hatch) lies directly between the corresponding compartments. Long corridors were split into virtual subcompartments to improve the accuracy of distance calculation, and some outdoors areas were included as virtual compartments. The ship plan we considered had about 300 compartments and 300 gateways.

The cost of traversing a path in this graph is intended to predict the time to follow it in a crisis situation. The cost is based on several factors that were defined and parameterized after discussions with experts on board a frigate. The primary factor is proportional to the distance between compartments along a path:

$$d_{12} = ((x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2)^{1/2}$$

where (x_1, y_1, z_1) and (x_2, y_2, z_2) are the coordinates of the representative points of two compartments. This is multiplied by two constants v (the “speed of people”) and K_{cij} (the compartment-status multiplier) to get the cost. This v was assumed constant over personnel, but a more sophisticated model could distinguish rates of different kinds of people. Smokey and flooded room multipliers (k_s and k_f, respectively) are applied to half of the distance between compartments d_{12} since on average, the gateway is midway between the compartments. (Smokey and flooded compartments can also be excluded from the graph at the choice of the operator during setup.) So the cost related to compartment status C_{CS} is:

$$C_{CS} = ½ \, v \, (K_{ci} + K_{cj}) \, d_{12}$$

where $K_{ci} = k_s$,if compartment i is smokey, k_f if compartment i is flooded, k_s k_f if compartment I is both smokey and flooded, and 1 otherwise.

Additional cost factors are defined for gateways. To reflect the expected additional time that will be spent in their usage. Parameter t_g measures the time to traverse the gateway when open (which is larger for narrow hatchways), to which is added a parameter t_c if the gateway is closed, representing the additional time to open the hatchway and close it after passing through. A cost C_{dc} is assigned for casualty-removal to gateways between decks of the ship (i.e. stairs and ladders) since it is hard to transfer a casualty from one deck to another. A cost C_{sp} is added for some gateways to model internal policies (as when orders are given to avoid a door, or when environmental conditions do not permit travel outdoors) or other miscellaneous factors (as when a door is broken). In total, the gateway cost C_G is:

$$C_G = v \, (t_g + t_c) + C_{dc} + C_{sp}$$

Then the total cost of an edge C_T is:

$$C_T = C_{CS} + C_G$$

When an operator requests a route from our system, costs for each edge are calculated from the current sensor information. For instance, sensors will tell us compartments that have smoke and whether certain important doors are open. Our system assumes reasonable default values when sensors are absent or nonfunctioning, to provide a degree of robustness. The modernization of the Niteroi class is providing additional sensors that we can use.

If the request is for a smoke-extraction route, a smaller graph is built where the nodes represent sealed sectors of the ship. Sealed sectors are bounded by gateways with sealed closings. The smoke will flow through a sequence of sealed sectors as it is being removed. The cost attributed to this traversal is the distance between centers of sectors plus a factor that reflects the difficulty of smoke passing through that gateway, as estimated by our experts.

5. Cost-minimizing Search

Once the ship is fully represented by a graph, we need a search algorithm to find the lowest-cost path. There are two basic search types, *blind* or *random* search and *direct* or *knowledge-based* search. Blind is a systematic search that uses no knowledge of how close we are to a solution in picking the path to follow our current location to a new location. In direct search we have some information that provides the best new location to choose.

Because the number of nodes of the ship was limited, we used the blind-search method of *branch-and-bound* for our task. (On a larger ship, the A* algorithm would be more appropriate.) Branch-and-bound at each step extends a path from the most promising node in the solution tree regardless of where this node is in the tree [Gonzalez and Dankel, 1993]. If a subsequent better route is found to a state, the route to there is updated.

As we implemented branch-and-bound in CLIPS (based on the Prolog implementation of the A* algorithm in [Rowe, 1988]), it uses these kinds of input facts:

(edge node1 node2 cost) - defines the edges of the graph
(start node-0) - defines the start node of the route
(end node-N) - defines the end node of the route

It produces the auxiliary facts:

(agenda node cost-to-here backpointer) - stores information for nodes to which routes have been already found, but from which subsequent travel has not been analyzed, together with the best cost found there so far and the predecessor node

(bestagenda node cost-to-here backpointer) - stores the current best agenda fact

It also produces output facts:

(oldagenda node cost-to-here backpointer) - stores the cost of the best route found to a node, together with the name of the predecessor node on that best route

Search begins with a single agenda fact for the starting compartment. Search ends when the best state on the agenda is the goal compartment. Then the backpointers are followed to find the route back to the starting compartment; this list is reversed and returned as the answer. If a route to the goal compartment cannot be found, a special answer is returned.

Our CLIPS implementation produces an answer in a very acceptable amount of time. In the worst case, including the graph construction and the translation of the room list to natural language, the program needs about four seconds to return the answer in a 400MHz Pentium II platform.

6. Application Module Descriptions

We now describe how requests are made and how answers are given to the operator for both application modules of the decision-aid system.

The CLIPS code is easily integrated to the rest of CAVFRAG system program since CLIPS is written in C. The whole system runs under QNX real time operating system which is based on message interchange. An interface between CAVFRAG Monitoring System and the CLIPS program exchanges:

- Status changes of sensors (to permit updating the knowledge base);
- Requests for the personal-route and smoke-extraction modules; and
- Calculated best routes and plans.

6.1 *Route Generator for People and Casualties*

6.1.1 *Start and End Compartments of the Route*

Starting and ending compartments for the route are obtained through a mouse selection sequence on a graphic display showing the ship plan of all decks, part of the basic CAVFRAG Monitoring System. “Waypoint” compartments that must be visited between start and end compartments can optionally be specified; then the program solves a set of subproblems and combines their results into a single path.

6.1.2 *Route Restrictions*

When starting and ending compartments are selected, the operator must fix details of the search by answering a set of questions in the monitoring-system window:

- Is it a route for casualty removal?
- Must it comply with designated one-way paths?
- Can it use scuttle hatches?
- Can it use outdoors areas?
- Can it use smokey rooms?
- Can it use flooded rooms?

These restrictions affect the graph construction (by eliminating some edges) and cost assignment.

6.1.3 ***Successful Searches***

The solution returned by the route-planning module for a successful search is a sequence of compartments and gateway codes. This is converted into a set of instructions in simple language to facilitate understanding by personnel.

6.1.4 ***Unsuccessful Searches***

A helpful expert system must be able to explain why it cannot find an answer when that happens. So our system does additional reasoning whenever a route to the ending compartment cannot be found. The "oldagenda" facts then hold all best paths found to compartments, and we save them in a new data structure. We then redo the search assuming that no restrictions (smokiness, floodedness, locked gateways, indoor travel, etc.) apply to compartment transitions. This new search must find a solution path because all compartments of the ship are connected. We then compare the "oldagenda" facts associated with the solution path to the stored "oldagenda" facts. The problem must lie at the first place in the path where the first search could not continue. We examine this place and summarize the problem for the operator in natural language. For instance, "The route is blocked between compartments A and B because compartment B is outdoors and you specified indoor travel." The operator can then modify their restrictions and rerun the search.

6.2 ***Smoke-Extraction Procedure Generator***

The smoke-extraction module is told by the CAVFRAG Monitoring System the compartment from where the smoke must be removed. This is done by mouse selection on the ship plan on the screen. The module then finds the sealed sector this compartment belongs to and searches for possible solution plans. A plan must specify:

- The smoke channel, the sequence of sealed sectors through which the smoke will flow.
- The gateways that must be opened and the gateways that must be closed to create the smoke channel.
- The fan and extraction devices necessary for this task.
- Suggested ship maneuvers (such as turning the ship with respect to the wind to facilitate smoke extraction).
- The sequence of actions that must be followed to accomplish the plan.

To generate a plan, the system uses a knowledge base of extraction procedures (basically, Standard Operating Procedures) created by ship experts. The procedures are organized by sectors and have associated conditions of use. We do a branch-and-bound search to find the proposed extraction route, the smoke channel. This requires finding the sector of the smoke-filled compartment and searching for a route to outdoors on a graph of sectors rather than compartments. This is a smaller search graph than for routes of people but it still requires a nontrivial search.

Each sector has several procedures for desmoking compartments in it, and the procedures are tested in order of complexity to find the first one whose conditions apply. Conditions include those on the availability and operability of equipment and the closability of gateways on the smoke channel. When a plan is found, a description in natural language is provided of:

- Ship maneuvers suggested;
- Doors, hatches, and scuttle hatches to be closed;
- Doors, hatches, and scuttle hatches to be opened;
- Devices to be used and where; and
- Other remarks.

Otherwise, the program will inform the operator that it cannot suggest any procedure to remove smoke from that sector.

7. Discussion and Conclusions

Expert-systems software appears to be an appropriate way to find people-removal and smoke-extraction routes on medium-sized ships. The CLIPS expert-system tool showed itself to be easy to use, and it provided clear and easy-to-maintain programs where the expert's knowledge was represented in a very direct way. In several cases the program obtained routes that were considered by our experts to be better than those used normally used by ship personnel during emergency drills. This induced some re-evaluation of current procedures on the ship. The system appears well accepted by users as it integrates well with the existing software for monitoring of ship operations. We hope to eventually develop similar onboard decision-aid modules for subsystems such as those for weapons and machinery control.

8. References

[Gonzalez and Dankel, 1993] Avelino J. Gonzalez and Douglas D. Dankel, *The Engineering of Knowledge-based Systems, Theory and Practice*. Prentice-Hall, 1993.

[FEMA, 1999] U. S. Federal Emergency Management Agency (FEMA), "Developing Effective Standard Operating Procedures for Fire and EMS Departments", Technical Report, 1999, available from http://www.fema.gov.

[Giarratano, 1998] Joseph C. Giarratano, *CLIPS User's Guide*, Version 6.10, U.S. National Aeronautics and Space Administration (NASA), 1998.

[New York Times, 2000] New York Times, "A Computer to Outsmart a Raging Fire", *New York Times*, March 7, 2000,

[Rowe, 1988] Neil C. Rowe, *Artificial Intelligence through Prolog*. Prentice-Hall, 1988.

[U.S. Navy, 1999] U.S. Navy, "Naval Ships Technical Manual", Volume 1, Chapter 55, "Surface Ship Firefighting", 1999.