

A Threat Ranking Algorithm for Multiple Intelligent Entities in a Simulated Environment

Ilker Gumus

GE-Harris Railway Electronics LLC.

PO Box 8900

Melbourne, FL 32902

igumus@ge-harris.com

Avelino J. Gonzalez

ECE Dept. University of Central Florida

PO Box 162450

Orlando, FL 32816-2450

gonzalez@ucf.edu

Abstract

It is often of utmost importance for a person to be able to determine which of several threats represent the most danger in order to react to it first. A mistake can result in damage, injury or in death. Furthermore, actions taken to address one threat, may place the person in a weaker position against other concurrent threats. This paper describes a methodology that uses heuristics to rank the various threats in order of increasing importance. The most important threat is then addressed through a proposed course of action that eliminates the threat. It then performs a simulation of the most likely outcome of the proposed course of action taken, to determine whether anything has changed with respect to the other threats. This validates the proposed course of action and permits it to be implemented. On the other hand, if the results of the simulation are such that the proposed course of action creates more problems than it solves, then it is discarded, and a new course of action is proposed. A testbed is developed and tested extensively to determine the viability of the concept.

1.0 Introduction

Training simulators need to represent many different entities that depict intelligent behavior in order to be considered useful under a variety of different missions. However, human control of entities is a formidable and expensive task, making it essential that some of the entities be controlled by computers. The reduction in the use of manpower decreases the cost of training and increases overall use of the training system, which translates into more experience and training for the trainees. In a simulator used for military applications, computer controlled entities are called *Computer Generated Forces* (CGFs). CGFs are a good way of representing friendly, opposing or neutral forces within a simulator.

There are several different methods used for controlling CGFs. *Semi-Automated Forces*

(SAFORs) are CGFs that operate under partial human supervision. An *Autonomous Intelligent Platform* (AIP), on the other hand, is an intelligent entity defined as a completely automated CGF. They are also, but less commonly, called *Automated Forces* (AFs).

SHORAD (Short Range Air Defense) Trainer, ASW (Anti Submarine Warfare) simulations, TRIO (An expert system for designed for teaching F-14 radar operations), SAFORs used in CBS (Combat Battle Simulation) system, IST-SAF (Institute for Simulation and Training Semi-Automated Forces), ModSAFs (Modular Semi Automated Forces) and CCTT-SAF (Close Combat Tactical Trainer Semi-Automated Forces) all fall under the category of SAFORs, even though they did have some different features that distinguish them from one another. [Brown 1994; Proenza 1997]. *State Operator and Result* (SOAR) / *Intelligent Forces* (IFOR) represents the first generation system with independent, intelligent, and fully autonomous AIPs used within various simulators to represent CGFs. The systems described above do not formally address the problem of threat ranking from the perspective of the AIP.

One of the most difficult tasks an AIP faces is behaving intelligently when it has to interact with multiple threats. The AIP needs to be able to sift through a large volume of information and be able to determine which threat is important and which is not. Being able to select relevant information from a large amount of stimuli is the key towards detecting critical situations. Although this task is a rather simple one for humans, it is not trivial for a computer. Unlike humans, computers are designed to handle very specialized tasks, one at a time. AIPs need a system that will allow them to function successfully and effectively when faced with several threats simultaneously. A method that offers a solution to

these problems is essential in achieving accuracy of behavioral representation for the AIPs. This paper addresses these issues.

2.0 The Solution Approach

We base our modeling infrastructure on the Context Based Reasoning (CxBR) paradigm. This paradigm has been shown to work effectively as well as efficiently when used to represent intelligent entities in a simulation. Refer to [Gonzalez et al 1994] for a description of CxBR.

In order to solve the problem of multiple event management, each AIP has to be able to prioritize, and in some cases even ignore, the stimuli it receives. For example, during maritime navigation in traffic, the AIPs are expected to ignore vessels that are far away and going away from them, as any such vessels do not pose immediate threats. The AIP is expected to focus on the most immediate potential hazard. For example, a close-by, large super tanker coming towards the AIP at full speed will require immediate attention. Such an event should take precedence over a small fishing vessel, far away, and moving away from the AIP.

The threat ranking system is composed of several steps. Stimuli collected from the environment surrounding the AIP are filtered through each of these steps. At the end, the system focuses on the most important hazard and takes appropriate action to avoid it.

1. **Determine what is and what is not a threat.** This is done based on the criteria that are most appropriate to the situation at hand. For example, while driving a car, you would not be concerned with vehicles very far away and/or moving away. On the other hand, a large truck closing in at full speed will definitely seize your attention.
2. **Prioritize the threats.** This step determines which threats need to be addressed first. From the driving example above, the nearby truck coming towards you at full speed will always have greater priority than an ambulance at a long distance away.
3. **If set not empty: From a set of possible actions to address the most important threat. Select the first or the most appropriate possible action. Then flag the action.** The reaction set of actions will be a finite set. Using the driving example above, the driver of the car could try several different actions: accelerate, decelerate, or maneuver away. Flying away, although not possible for a car, would be an alternative for a bird faced with a similar scenario. The set of possible actions can be ordered in terms of what actions should be investigated first. Ordering the

sets of possible actions will also enable the incorporation of individual characteristics into AIPs. This way, faced with the car/truck scenario described above, some AIPs will be more likely to turn away while others will try to speed up and get out of the way. Flagging is done in order to avoid selecting the same action again within the loop.

If empty: Go to emergency action, then go to step 1. Reaching this step means that all possible actions that might provide a solution to the problem have been attempted, and no acceptable solution can be found. In such cases, a prescribed emergency action will be taken. These will be the types of actions that can be used as a last resort. The emergency action will also be determined based on the situation at hand.

4. **Check if the action selected results in a compromising situation with the other threats.** What constitutes a "compromising situation" should be determined based upon the situation at hand. In the car example, you would not want to get hit by another car while trying to avoid a turtle or a possum on the road. How many steps ahead should be considered is also based upon the situation at hand and the environment. Some environments may require detailed planning, (i.e., battle scenarios, navigation through a maze.) while others may require considering a maximum of a few steps. (i.e. driving in a highway.)
5. **If the action under investigation does result in a compromising situation, then go back to step 3.** If the possible action does not solve the problem or causes bigger problems, then try the next possible action on the list.
If not: Execute the action and go to step 1. The situation is resolved. Now it is time to go back and look out for other possible problems or threats.

The system described above is very general and requires some parameters to be set in order to be useful. Since this project involved the implementation of this algorithm for simulating appropriate behavior of nautical vessels, the heuristics used for this purpose will be discussed next. The object of each behavior is, of course, safe navigation to destination while avoiding collision.

The definition of the environment will determine how the algorithm above will apply. The algorithm is designed to be very general. It is intended to work for all intelligent entities in all kinds of different situations. Different definitions of the environment will allow the algorithm to work for automobiles in traffic simulations as well as in a simulation of wild animals in an African savanna. It

should be remembered that the algorithm is written from the point of view of the AIP itself. The definition of the environment also includes the definition of *self* (self is the AIP that is controlled) and the definition of entities with which self has contact. Therefore the definition of self will determine sets of possible actions, sensitivity to stimuli, and quickness of reactions.

3.0 Application Domain

A prototype was built to evaluate the concepts presented above. This prototype represents the behavior of one vessel that interacts with others. Each AIP has the Nautical Rules of the Road embedded within itself, as the basis of conduct. The Nautical Rules of the Road are a set of rules that all vessels navigating in public or international waters must obey. They are similar to the traffic rules that drivers must obey in roadways. The embedding of the rules of the road into AIP architecture is accomplished through the use of contexts. The top-level context is the Mission Context (Gonzalez 1994) or, in short, the *mission*. Since each AIP is traveling to some location, each AIP has a similar mission.

The system is composed of two different parts, C++ and CLIPS. Most of the procedural and mathematical operations within the contexts, such as distance and velocity calculations between two vessels, are done in C++. On the other hand, most decision-making parts that incorporate heuristics, such as the implementations of the rules of the road, are done using CLIPS. Contexts are implemented as transparently and as flexibly as possible over the two languages. No particular context is completely implemented in one particular language.

For this project, the environment is a maritime environment where surface vessels navigate. In our case, self would be a surface vessel. Surface vessels can be super tankers, large commercials, small commercials, fishing boats, sailing vessels, unsubmerged submarines, power boats and any other kind of vessel that floats. The entities that self will have interaction with would be other surface vessels, which will include all vessels listed above.

The Scenarios

The environment is further divided into distinct but general scenarios. These will form the basis of contexts. The definitions of each scenario will affect actions and behavior first hand, because the selection of actions will depend on the recognition of each scenario. It should be noted that, each scenario requires totally different sets of actions that will determine appropriate behavior. For our purposes, the

scenarios will be divided into two: open-water and channel.

Possible Set of Actions

Sets of possible actions, define what an AIP is capable of doing. Sets of possible actions will depend on the AIP represented. Since this project deals with ships, the possible sets of actions for an AIP representing a nautical vessel are the following:

1. *Increase or decrease speed.*
2. *Turn to port or starboard.*
3. *A combination of 1 and 2.*

No actions may be taken outside of what is defined. Unless stated otherwise, the vessels cannot fly or dive in order to avoid collision.

Environmental heuristics are set of small pieces of information that greatly affect the behavior of each AIP. These heuristics are scenario-dependent. They will be used mainly to determine what is not a threat. The following are heuristics relevant to the marine environment and to all other nautical definitions described so far. These will be implemented in order to add realism to the behavior of the AIPs.

The following heuristics were deemed appropriate for channels:

1. *Any vessel that is not bound by the channel is not an immediate threat under normal conditions: This includes all crossings. Exceptions are the situations where emergency action is required as per Rule 17. According to the nautical rules of the road the vessels that are not bound by the channel are deemed as "give way". Therefore, taking appropriate action is the responsibility of that vessel, unless there exists a serious threat of collision.*
2. *All vessels that are lighter or smaller are not a threat under normal conditions. Unless there exists an immediate threat of collision, lighter vessels are considered to be in give-way status. This puts the burden of avoidance on to the smaller/lighter vessels.*
3. *All vessels that are engaged in non-threatening activities are not a threat. In this case, vessels going away from the vessel itself, or to put it another way, when the distance between the vessels is increasing.*

Threat Ranking Heuristics

Threat ranking heuristics determine which threat should be treated first. They determine the most immediate hazard based on the environment, the scenario and the situation at hand. Due to the necessity of numerically ranking the hazards in order to determine the greatest danger, a *threat index* function was used. The parameters that such a

function takes into account will be determined by the threat ranking heuristics.

1. *The nearer, the faster, the bigger a ship is, the more of a threat it is.* In the case of an AIP in a nautical traffic environment, a large, fast and near vessel coming in your direction has a greater potential for destruction. Therefore, such vessels deserve a high priority.
2. *The smaller the time to collision, the greater the threat.* A vessel that is first to potentially collide presents the greatest threat. This parameter, time to collision, should also be taken into account.

The threat index function (tif) was used to determine the rank and the order of importance of each threat. The function that was used for ranking takes parameters such as speed, distance, weight and time and computes a threat index. The function also has four adjustable weights. This way some of the criteria can be weighed more heavily than others. This will help balance each value and reduce possible numerical biases. The function is the following:

$$tif = \alpha \text{ mass} + \beta \text{ speed} + \gamma (1/\text{distance}) + \phi (1/\text{time})$$

Mass: The mass of the ship (tonnage.).

Speed: The speed of the vessel in knots.

Distance: The distance from the vessel to the AIP in nm.

Time: The time to closest point of approach in seconds.

For the software being developed, values for α , β and γ were set to be 0, due to the fact that in the test bed used the most influential, and probably the only, factor is the time to collision. The value for ϕ was set to be 100 in order to avoid clumping of values. These values can be adjusted in order to fit the requirements of the AIP. The values for these parameters should be assigned by the experts in the field. These values can be adjusted individually for each platform in order to represent character traits such as aggressiveness and shyness. The larger a threat index, the greater a threat is.

The threshold for threat index will allow some of the threats to be ignored. By using a value like this, characteristics for captains such as carelessness and carefulness can be incorporated. The threshold values for the test bed were set to zero, which means all of the inputs are considered. The paradigm, however, does allow room for varied values.

The system described above can be broken down to the threat ranking algorithm below:

1. Read the list of vessels from the simulator.
2. Until the list of vessels is empty, do the following:

- 2.1 Check if the first vessel on the list is navigating the channel.
- 2.2 If yes do the following:
 - 2.2.1 Calculate its threat index using function
 - 2.2.2 Check if its threat index is greater than the maximum threat index
 - 2.2.3 If yes, set this vessel as the most important threat.
 - 2.2.4 If not, remove this vessel from the list.
- 2.3 If not, remove this vessel from the list.
3. Do situational assessment and determine the type of the encounter according to the nautical rules of the road. (i.e. head-on, overtaking, overtaken, crossing.)
4. Based on the type of the encounter, determine if the situation requires give-way or stand-on status.
5. Based on the status do the following:
 - 5.1 Read the list of possible actions.
 - 5.1.1 Until the list of possible actions is empty do the following:
 - 5.1.2 Check if the first action resolves the situation.
 - 5.1.3 If yes, check if the action results in a compromising situation with other threats.
 - 5.1.3.1.1 If yes, remove that action from the list.
 - 5.1.3.1.2 If not, execute the action and quit the algorithm.
 - 5.1.4 If not, remove the action from the list.
 - 5.1.5 Switch to the emergency action and quit the algorithm

As required by the algorithm, the AIP structure operates under a loop. This main loop tries to encompass the elements described within this chapter. The main loop is composed of the element below:

Loop:

Update Information
Asses Situation
Consult Expert System
Select and Execute Context

The *Update Information* element basically looks around and notices other platforms and elements around the AIP. This portion is the senses of the AIP. It bombards the AIP with all kinds of information.

All the information coming from Update Information is fed directly into Filter Stimuli. The threat recognition algorithm is applied here. This element filters out the information that is not of concern to the AIP, or in this case, does not pose a threat in any way. The algorithm is applied until the most important threat is found. This information is passed on as to the next element.

Assess Situation looks at the situation at hand and classifies this as one of the previously abstracted contexts. In the test bed used, classifications are based on the type of encounters as defined in the nautical rules of the road.

The *Consult Expert System* element feeds all the information gathered so far to the expert system. The expert system provides tactical information to the AIP.

Select and Execute Context selects a context based on the information and feeds all the required information into the context chosen. The context takes control of the AIP and controls it until the context expires. The loop is re-started if a context expires or is interrupted by an emergency.

4.0 Prototype Evaluation

An extensive set of tests were carried out in order to ensure that the algorithm is capable of performing the tasks it is expected to do. The tests were performed on the software implemented as the test bed, discussed in section 3.0. The tests and the results are described in this section.

The tests were broken down into several portions. In order to test the behavior of the AIPs in increasingly complex situations. First, the AIPs were tested on one-to-one encounters. This means that the self AIP is faced with only one other vessel. Then the AIP was subjected to one-to-many tests. This means that the self AIP operated in an environment where there were more than one vessel, albeit unintelligent. Last, the AIPs were subjected to many-to-many tests. This means that the self AIP was now put in an environment where there were several other intelligent vessels around it.

Summary of Test Results

The tests proved that the solution approach is effective. All of the test results were correct and consistent according to the nautical rules of the road, both in the channel and in the open-water contexts. Refer to [Gumus 1998] for a detailed description, as

space limitations preclude their inclusions here. The software behaved as expected: correct and consistent according to the nautical rules of the road. The varied nature of the tests, such as testing different speeds, under different contexts and scenarios shows the robustness of the solution approach. This proves that the threat ranking algorithm is a sound, useful, and effective method for representation and management of multiple entities in simulated environment.

5.0 CONCLUSIONS

The work described here establishes a method for modeling acceptable behavior for an AIP existing in a simulated environment, in the company of many potentially threatening and non-threatening stimuli. The algorithm designed to accomplish this task was reduced to code in order to represent intelligent behavior of nautical platforms according to the Nautical Rules of the Road. The test performed proves that the algorithm designed and the paradigm used are sound methods for accomplishing the desired goals.

References

[Brown, 1994] Brown, J. C., (1994). Application and Evaluation of the Context-Based Reasoning Paradigm. Master's Thesis, Dept. of Electrical & Computer Engineering, University of Central Florida, Orlando.

[Gonzalez, 1994] Gonzalez, A. J., Ahlers, R. H., (1994). "A Novel Paradigm for Representing Tactical Knowledge in Intelligent Simulated Opponents. *Proceedings of the Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Austin, May 31 - June 3.

[Gumus, 1998] Gumus, I., (1998). A Threat Prioritization Algorithm for Multiple Intelligent Entities in a Simulated Environment. Master's Thesis, Dept. of Electrical & Computer Engineering, University of Central Florida, Orlando.

[Proenza, 1997] Proenza, R., (1997). A Framework for Multiple Agents and Memory Recall Within the Context-Based Reasoning Paradigm. Master's Thesis, University of Central Florida, Orlando.