Image Cover Sheet

CLASSIFICATION	SYSTEM NUMBER
UNCLASSIFIED	149640

TITLE

GENERAL PRINCIPLES IN THE DESIGN OF REAL-TIME KNOWLEDGE-BASED SYSTEMS FOR THE THREAT EVALUATION AND WEAPON ASSIGNMENT PROCESS

AN: 95-00582

System Number:

Patron Number:

Requester:

Notes:

DSIS Use only:

Deliver to: NL

General Principles in the Design of Real-Time Knowledge-Based Systems for the Threat Evaluation and Weapon Assignment Process

Robert L. Carling[1]
DREV[2], P.O. Box 8800,
Courcelette, Québec, G0A 1R0.

Abstract

The threat evaluation and weapon assignment (TEWA) process is an important process in the command and control system of an anti-air warfare (AAW) destroyer or frigate, since it ranks the air threats detected by the ship's automatic track management system (ATMS) according to specific criteria depending on the threat and the ship's characteristics and then assigns a hardkill or a softkill weapon to these threats in order to destroy or decoy them. A nonreal-time knowledge-based system has been built in SMALLTALK 80/HUMBLE to do threat evaluation and weapon assignment for a single stationary AAW destroyer attacked by anti-ship missiles. The knowledge-based system has been further refined to produce TEWA decisions for a manoeuverable ship attacked by anti-ship missiles.

This paper describes the algorithms of the TEWA knowledge-based system that integrate the use of hardkill/softkill weapons with the ship's movement functions.

This paper presents the real-time requirements of a knowledge-based system integrated into the air combat system of an AAW frigate. The high and low level designs of a real-time knowledge-based system for TEWA in terms respectively of partitioning artificial intelligence (AI) systems and assigning tasks are also presented. A transputer farm implementation of the threat evaluation function and a mapping of weapon assignment knowledge bases onto transputer networks are also described in qualitative terms. An alternative real-time implementation of AI tasks using Reduced Instruction Set Computer (RISC) processors and the associated problems of multiprocessor scheduling are also discussed.

Introduction

The design of the knowledge-based system for the TEWA function of a stationary AAW destroyer or frigate attacked by anti-ship missiles has been documented in references [1], [2] and [3]. The knowledge-based system comprises four knowledge bases called the Threat Evaluation knowledge base, the Result Evaluation knowledge base, the Force Resources Evaluation knowledge base and the Candidate Reaction Evaluation knowledge base. The first knowledge base comprises about fifty rules, while the remaining three knowledge bases together contain about fifty rules. The knowledge bases use backward chaining, multiple confidence factor paradigms, temporal reasoning and spatial reasoning. References [1] to [3], describe how threat ranking values are calculated from identity, kinematic, radar state, and engagement status parameters using the HUMBLE uncertainty calculus. They also describe how resource allocation plans are formed for hardkill and softkill weapons, how these plans are ranked and then how they are implemented against anti-ship missile threats. These knowledge bases and the plan formation and ranking will be known as the high level design of the knowledge-based TEWA.

A multiple ship anti-air warfare simulator is now being developed in SMALLTALK 80, in which each ship has a knowledge-based TEWA and the ships may manoeuvre before deploying hardkill or softkill weapons. The design of the anti-air warfare simulator is presented in reference [4].

Neither the initial knowledge-based system nor the multiple ship anti-air warfare simulator are real-time simulations. A resource loading study was undertaken to measure the effect of replacing the conventional TEWA in a TRUMPlike ship by artificial intelligence functions. The command and control computers, the AAW functions, the SHINPADS bus and the LINK 11 system are modelled in reference [5], while the artificial intelligence functions replacing the TRUMP TEWA are described in reference [6]. A Simscript model was devised to simulate anti-ship missile attacks on convoys of TRUMPlike ships. The simulation models the flow of messages from the sensors and weapons to the command and control computers and it also models the queue sizes of tasks arriving in the interface of each command and control computer. This model is described in reference [7]. Results from scenarios in which a small number of anti-ship missiles attacked three TRUMPlike ships and

[1] Phone (418) 844-4432.

[2] DREV = Defence Research Establishment Valcartier.

a very large number of anti-ship missiles attacked each ship of a four ship convoy have been obtained. In the latter scenario, there were very large input and output queues to the command and control computers of all ships in which AI functions had replaced the conventional TEWA. It, therefore, was concluded that faster processors and different architectures would be required in order to reduce the input and output queue sizes of command and control computers executing AI functions. The two possible candidates which could conceivably replace the current processors on board the TRUMPlike ship are transputers or RISC processors.

TEWA Reactions for a Manoeuverable Ship

The AAW simulator is being developed for several warships which can manoeuvre when attacked by anti-ship missiles. Ship rotation will be used to support hardkill reactions. More specifically, ship rotation may occur in order to unmask a fire-control radar or naval gun that is in a blind zone.

In order to design a TEWA that supports rotation before hardkill deployment, the following steps will be taken. The reactions that are considered unengageable because there are blind zone problems will be noted. The amount of rotation required to make each reaction engageable is calculated. An attempt is made to create engageable reactions before considering hardkill reactions that involve rotation. A certain number of penalty points are associated with the plan when a component reaction involves a rotation because of the delay incurred before it can be implemented. Whenever, a plan P has a constituent reaction R that incorporates a rotation, calculations are made to determine whether other reactions in the same plan which are engageable would be made unengageable as a result of the rotation. Now, consider a plan P = [R1, R2, R3] where R1 is a reaction that involves a rotation while the others do not involve rotation. Further, assume that P has a score of X according to the existing methods of scoring plans. Suppose that it is determined that the implementation of R1 would make R2 unengageable. In such a case, the plan P is modified to be [R1, R3] and a new score is calculated including the previous penalty factor. Consider a plan P = [R1, R2, R3] where R1 and R3 both require rotation. It is always possible that plans may be drawn up (e.g., plan P) that have more than one reaction involving rotation. Clearly, both rotations cannot be implemented. In this case, the rotation is chosen which is associated with the reaction assigned to the highest ranked threat.

With the current TEWA design of the AAW simulator, precedence is given to the implementation of hardkill reactions including those needing rotation before the implementation of softkill reactions. The most general form of softkill reaction which is typical of chaff, jammers and passive decoys is a rotation of the ship to place it in the best position for softkill weapon deployment followed by a change in ship velocity to ensure feasibility of the softkill weapon deployment. After the weapon is deployed, another rotation and change in ship velocity give the ship the best chance to leave the threat-weapon engagement zone without any damage. In the last example given above, i.e., in the plan P consisting of [R1, R2, R3] where both R1 and R3 require rotations, R3 could be a hardkill reaction and R1 could be a softkill reaction. Thus, R3 is assigned to the most highly ranked threat and if the effect of implementing R3 does not prevent the softkill reaction R1 from being implemented, it will also be executed by the ship.

The Real-Time TEWA Problem

Many designers of successful real-time artificial intelligence systems have adopted a hierarchical approach. The designers of the EAVE (Experimental Autonomous VEhicle) robot at the National Research Council adopted a real-time AI model in which a rule-based system provided the high-level decision making and decided in real-time on which actions the robot platform should take. The details of this design and platform task levels are given in references [8] and [9]. Reference [10] also mentions a systematic way of designing knowledge-based systems at two different levels. In this article, it suggests that many AI problems are really complex generic tasks, i.e., they are further decomposable into components that are more elementary in the sense that each of them has a homogeneous control regime and knowledge structure. The high level design of our knowledge-based system has been explained in the introduction. The three low level implementations of the TEWA that could conceivably execute knowledge bases in real-time are a single fast processor, transputer networks or distributed RISC processors.

The knowledge bases of the TEWA will have to interact with various combat system elements. In an air engagement, the TEWA will receive data from various sensors and LINK 11. The data coming from these sensors and LINK 11 can be modelled by synchronous or asynchronous processes and must be processed by the TEWA knowledge bases within a certain real-time constraint. As soon as the threats have been identified to be hostile, the TEWA knowledge bases must do engageability calculations within a certain real-time requirement and then send the engagement orders to the designated weapon. If softkill weapons are going to be used, then the TEWA must send a request for information to the EWCP (Electronic Warfare Control Processor) which replies to this request according to the real-time design of the

EWCP. There will be a real-time requirement for the total time required to send a request from TEWA to the EWCP, to have it processed by the EWCP and send an answer back to TEWA. In preparation for missile launch or gun firing and after missile or gun intercept with the target, there will be many messages sent from TEWA to weapon system elements and vice-versa. Each of these must be executed within a certain real-time constraint.

Up to the present moment, the author has not been able to find any document specifying upper bounds for the real-time requirements described in the previous paragraph. The only documents available are those giving the global reaction time of the ship from fire-control radar designation to surface-to-air missile (SAM) launch when the ship is faced with a quick reaction threat. In references [5] and [6], Thomson CSF has made estimates for the TEWA, sensor and weapon element processing times when messages are created, are relayed or terminate at these combat system nodes. These processing requirements were devised for moderate threat loads and for a TRUMPlike ship equipped with militarized processors. References [5], [6] could be used as a starting point for the TEWA real-time requirements, but these values may have to be decreased for heavier threat loads.

Loading Knowledge Bases onto Transputer Networks

The real-time implementation of the high-level TEWA design can be done so that some of the knowledge bases are executed concurrently and some are executed sequentially. An implementation can be attempted where the Threat Evaluation, Result Evaluation and Force Resources Evaluation knowledge bases are evaluated concurrently and then the Engageability, Effectiveness and Reaction Ranking of resource allocation plans are calculated sequentially after the three initial concurrent computations. By the very nature of engageability calculations, the two subprocesses of Candidate Reaction Evaluation (i.e., engageability and effectiveness) that generate resource allocation plans and the Reaction Ranking of these plans have to be sequential. The other way of implementing the TEWA process is to execute sequentially each of the knowledge bases Threat Evaluation, Result Evaluation, Force Resources Evaluation, Candidate Reaction Evaluation and the Reaction Ranking function. It is to be noted that although the Candidate Reaction Evaluation and Reaction Ranking processes are sequential, there is some inherent parallelism in them. When engageability and effectiveness are being calculated for hardkill and softkill weapons, they can be done concurrently, since they are completely independent of each other. At the present moment, with the current way that plan ranking takes place, it is feasible to do simultaneous reaction ranking by using duplicate reaction sets for a ship that does not rotate and for a ship that needs to rotate before deploying hardkill or softkill weapons. Since transputers may be used for concurrent and pipelined applications, some general steps will be presented which will have to be performed before any transputer implementation of knowledge bases can take place.

In order to make the Threat Evaluation process as rapid as possible for a large number N of tracks, the ideal situation is to try and make the time for calculating the threat levels for N tracks be equal to the time for calculating the threat level of one track since the N tracks in our current threat evaluation model are basically independent of each other. The Threat Evaluation problem has the characteristic that the same set of instructions (namely, the fifty rules of the knowledge base) are applied to each track. This kind of problem is known in the parallel processing literature as the "farming" problem and according to reference [11] is particularly well adapted for implementation in a network of transputers.

The Candidate Reaction Evaluation and Reaction Ranking processes are sequential with some inherent parallelism. Unlike the Threat Evaluation process, these two processes are not farming processes in the sense that the same set of instructions is applied to the incoming data. If an incoming track is not engageable, it is immediately discarded from the pipelined processes of Engageability, Effectiveness and Reaction Ranking. In the case of the Result Evaluation and Force Resources Evaluation knowledge bases, updates are sent to them each time the result of a reaction or the status of a resource are known and in this case only the pertinent rule is fired. Thus, these two knowledge bases also cannot be considered to be "farming" processes.

Reference [11] recommends the following steps for determining a transputer network which will satisfy the real-time requirements of the AAW problem. These are:

Step 1 : Functional decomposition
Step 2 : Partitioning and Allocation
Step 3 : Scaling.

These three steps could well be applied to the Threat Evaluation, Result Evaluation, Force Resources Evaluation and the Candidate Reaction Evaluation /Reaction Ranking knowledge bases. In the functional decomposition stage, it is necessary to identify the set of logical modules and data files making up a transputer implementation of the problem. If the Threat Evaluation problem is considered as a "farming" process, the master process must be identified along with the worker processes. One possible formulation is to model the master process as being the fused track database and the worker processes as separate copies of the Threat Evaluation knowledge base.

In the partitioning stage, the logical modules and data files that are the output of the functional decomposition are partitioned, i.e., are mapped into a set of parallel processes executing on their own data and on non local data exchanged with other processes by means of a suitable communication network. Two objectives now have to be met. First, the grain size of the partitioning into parallel processes (defined to be the amount of computation assigned to each process) must be maximized in order to obtain a short computation time for the relevant process. Secondly, if the grain size is too small, there will be large delays caused by interprocessor communication which will cause the total execution time to be large. Thus, a grain size has to be obtained which is small enough to keep the computation time fast but not so small that the communication times are too large. In the case of the Candidate Reaction Evaluation knowledge base, the fastest solution can often correspond to a partition with a physical meaning. In this case, there is a natural partitioning between reactions that are entirely hardkill, that are entirely softkill, involve rotation before hardkill/softkill deployment or do not involve rotation before hardkill/softkill deployment. In addition to determining the grain size of partitioning, the transputer topology must be chosen. The transputer topology is often chosen as a function of the physical problem . Many choices have been put forward such as ring topologies, square mesh topologies, and hypercubes (see reference [11]). However, references [12] and [13] suggest pipelined architectures of transputers for pipelined problems and any network (small or large) of transputers in which members are connected to each other by at most four communication channels for a general problem. The transputer network is usually implemented using a host computer and a root transputer, i.e., the tasks from the ship combat system are sent to the host computer and then to the root transputer that acts as a system manager routing each task to the appropriate transputer for execution. The topology that is chosen will obviously influence the kind of routing algorithm used for each task. Routing systems for transputer farms are explained in reference [13], p.205, while message passing from the root transputer to general transputer networks are given in reference [12], pp. 139-156. In general, the topology chosen should be compatible with the grain size, and the routing scheme chosen should be the most efficient for the given topology.

In general, according to reference [11], the problem of software allocation of n processes to m processors is NP complete. In this particular case, there may be more than one solution. The most important point is to find a solution that satisfies the real-time requirements for Threat Evaluation, for Result Evaluation, for Force Resources Evaluation and for Candidate Reaction Evaluation/Reaction Ranking and to prove that this solution satisfies the above real-time requirements.

After going through the process partitioning stage and the software to processor allocation stage, a design for a topology of transputers with an adequate communication system and with knowledge bases assigned to the transputers will have emerged so that the total execution time for each knowledge base is minimized for a large number N of tracks. However, it may still be slower than the real-time requirement. If this is the case, a scalability study will be undertaken. The goal is to find the number of processors that makes it possible to obtain a performance that is as close as possible to the desired one without making the system inefficient. The speed-up obtained by using a parallel architecture is not a linear function of the number of processors used: as the number of processors increases, the diameter of the network interconnecting them and the distances between communicating processing elements also grow. Furthermore, as the size of the problem is usually fixed, there is an inevitable reduction in the grain size. This means that, starting from a given system configuration, performance improvements due to the addition of a new processor become smaller and smaller, efficiency decreases and the cost/performance ratio grows.

In practice, for scaling to be performed correctly, it is important to determine the number of processors representing the threshold beyond which the speed-up is no longer linear and the computing resources are used inefficiently. In real-time systems, the objective is to obtain a given performance and it may, therefore, be reasonable to choose a high number of processors, even if used inefficiently. During the scalability study, it is also necessary to do a performance analysis of the transputer system. At present, the most widely used indices are speed-up and efficiency. The speed-up obtained for a certain problem by using a parallel computer is commonly defined as the ratio between the response time for that problem using a single processor and the response time when a parallel computer made up of K processors is used. If the latter of these times is indicated by $T_{par}(K)$ and the one measured under completely sequential conditions by T_{seq}, then obviously $T_{seq} = T_{par}(1)$. Using this notation, the speed-up S (which is, of course, a function of K) is given by the expression:

$$S(K) = T_{seq}/T_{par}(K)$$

Efficiency ε is usually used along with speed-up as a performance index, and is defined as follows:

$$\varepsilon = S(K)/K$$

This definition of efficiency presupposes that processors are the critical resource of the system. It is

the most accepted definition of efficiency and does not take into consideration any of the other important factors that contribute to efficiency such as : algorithmic overhead, software overhead, uneven load balancing, communication overhead and external communication overhead.

In order to study the above proposed mappings and their interaction with interrupts coming from the ship's combat system, a Petri-Net simulation can be used as in reference [14]. The results of some parallel processing studies show that good scalability can be obtained for certain architectures, transputer types and host computers but when these are changed the scalability is not so good (see reference [15]).

RISC Processor Implementations of Artificial Intelligence

The information flow from the ship's sensors, weapons and LINK 11 to the AI TEWA functions during an anti-ship missile attack can be modelled by periodic processes. However, some sensors such as ESM only update their state and hence send an interrupt to TEWA each time the external environment changes. In existing real-time fusion systems, the current practice is to associate the incoming asynchronous data with the appropriate cycle of periodic data (such as radar). As soon as the anti-ship missile threats become engageable by the hardkill and softkill weapons, the number of tasks received by the AI TEWA functions on the ship's processors increases. Suppose that the processors of the TRUMPlike ship are replaced by hypothetical RISC processors. One of the very important considerations determining how many functions will be loaded onto a RISC processor, and which ones will be loaded, is the scheduling of these functions on the multiprocessor architecture for the command and control system of the ship. If we assume that data coming from all sensor, weapon and Link sources is periodic, important results developed for the rate monotonic scheduling of uniprocessors made be used to schedule AI functions on multiprocessors.

The first important result concerning the rate monotonic scheduling of a single processor was discovered by Liu and Layland and published in reference [16]. Their theorem states:

Theorem(Liu and Layland)

A set of n independent periodic tasks scheduled by the rate monotonic algorithm will always meet its deadlines, for all task phasings, if

$$(C_1/T_1)+ \dots +(C_n/T_n) \le n(2^{1/n}-1) = U(n) \qquad (1)$$

where C_i and T_i are the execution time and period of task τ_i.

The above theorem offers a sufficient condition that characterizes schedulability of a task set under the rate monotonic algorithm. This bound converges to 69 percent (ln 2) as the number of tasks approaches infinity. The values of the scheduling bounds for one to ten independent tasks are as follows:

U(1) = 1.0, U(2) = 0.828, U(3) = 0.779
U(4) = 0.756, U(5) = 0.743, U(6) = 0.734
U(7) = 0.728, U(8) = 0.724, U(9) = 0.720
U(10) = 0.7177

References [17] and [18] explain how to use this theorem for simple examples and how the assumption on periodicity can be relaxed by the introduction of sporadic servers for aperiodic tasks. The algorithms for sporadic servers can be used to program interrupts for multiprocessor systems (see reference [19]). The condition of the Liu and Layland theorem is not however necessary as shown in reference [20].

Multiprocessor Scheduling

The basic idea of this section is to try and extend some of the scheduling results from a single processor to several processors in order to develop a static scheduling theory for multiprocessors. Rate monotonic algorithms are better suited to static scheduling of multiprocessors (each processor runs the same set of tasks throughout an air engagement) than to dynamic scheduling of multiprocessors (see reference [21]). The following theorem will be stated and some indications will be given of its proof:

Theorem 1

If $\{e_i\}$, i=1,...,k is a set of k periodic tasks and $\{E_i\}$ is a set of processors each having the same processing speed and for which the execution time C_i of a task is less than or equal to the arrival time T_i, then there always exists a minimal number m of these processors such that for each processor of the set $\{E_i\}$, i=1,...,m inequality (1) is true for all the tasks assigned to E_i.

Proof : The theorem may be proved by ordering the utilizations C_i monotonically, assigning the tasks to each one of the processors so that the Liu and Layland conditions are satisfied and then starting from the tasks assigned to the first processor systematically rearranging them among the remaining processors. Each time a rearrangement is successful the number of

processors decreases by one. Eventually, the process of reducing the number of processors executing all tasks will stop and this will give the minimal number m.

In many cases, T_i is smaller than the execution time C_i. In order to answer scheduling problems where $C_i > T_i$, the following theorem can be proved.

Theorem 2

If there are k tasks $\{e_i\}$ satisfying $T_i < C_i < 2T_i$, then m = 2k will give a set of processors that can schedule all tasks but k of these processors will be inactive for $n(2T_i - C_i)$ ms where n = ceiling$(T/2T_i)$ for some finite time interval T. Moreover, if the inactive time of the above m processors are used to schedule tasks satisfying $C_i > 2T_i$ then the deadline for completing these tasks occurs after $4T_i$.

Proof

If D_i is the deadline for completing task e_i, the fact that

$$D_i > (2T_i/T_j)2T_j = 4T_i$$

follows from

$$n(2T_j - C_j)/C_i > 1.$$

In view of theorem 2, it is better to execute tasks of long duration, i.e., tasks for which $C_i \geq 2T_i$ on a faster processor in order to satisfy its deadline. The next question that arises is how fast should this processor be. The following theorem will provide an answer to this question.

Theorem 3

Let $\{e_i\}_{i = 1,...,k}$ be a set of k tasks with execution times C_i and arrival times T_i on a set of processors E_i. Define

$$N = \max\left(5, \max_{i = 1,...,k} \text{ceiling}(C_i/T_i)\right) \qquad (2)$$

If F_i is a set of processors that can reduce the execution times of the tasks e_i to C_i/N, then
(1) there exists a minimal set of processors of type F_i for which the Liu and Layland conditions are satisfied,
(2) the number of processors of type F_i is considerably less than the number used in theorem 2,
(3) all tasks for which $C_i > 2T_i$ will satisfy their deadlines, i.e., $C_i/N \leq T_i$.

Proof

The three above statements can be deduced from equation (2).

Example:

The following data was taken from a simulation where several anti-ship missiles attacked a TRUMPlike ship. The execution times C_i and arrival times T_i of tasks for all AAW functions are given in milliseconds. The TEWA design, i.e., the distribution of AAW functions to processors was chosen arbitrarily and is not necessarily ideal.

AAW function	C_i (msec)	T_i (msec)
CPCI1		
Navigation I/F	1.00	14.0845
Rapidly Firing Gun I/F	4.62	28.169
Fire-Cntrl Radar I/F	11.20	9.337
GunFCR I/F	11.20	15.29
Target Data	7.89	3.3647
CPCI2		
ATMS I/F	3.60	29.112
Target Evaluation	41.35	69.444
CPCI3		
WDS I/F	9.37	6.4226
Result Evaluation	22.50	15.987
Candidate Reaction Evaluation	25.37	15.72327
Ranking of Candidate Reactions	6.96	25.00
Reaction Selection	3.09	10.288
CPCI4		
LINK 11 I/F	9.46	5.68182
CPCI6		
ESM I/F	40.20	9.19118
CHAFF I/F	1.01	30.30303
Force Data	1.92	2.66064
Forces Resources Evaluation	7.50	14.9365198

The utilizations of the seven functions for which the execution time C_i was greater than the arrival time

T_i were calculated in the ship data problem and are given below.

AAW Function	Utilization($x_i = C_i/T_i$)
Fire-Cntrl Radar I/F	1.199529
Target Data	2.344934
Weapon Direction System I/F	1.4589107
Result Evaluation	1.4073935
Candidate Reaction Evaluation	1.613532
LINK 11 I/F	1.6649595
Electronic Support Measures I/F	4.373758

When theorem 3 is applied to the seventeen AAW functions of the example, and each of their task execution times is divided by 5, the following assignment of tasks to the processors $\{F_i\}$ is obtained.

F_1 : .006666<.0142<.024732<.032802<.05568 <.06007<.100425<.119088876<.1443262 <.146501

$$\sum_{i=1}^{10} x_i = 0.704491 < U(10) = .7177$$

F_2: .239906<.2814787

$$\sum_{i=11}^{12} x_i = .5213847 < U(2) = .828$$

F_3: .29178214<.3227064

$$\sum_{i=13}^{14} x_i = .61448854 < U(2) = .828$$

F_4: .3329919<.4689868

$$\sum_{i=15}^{16} x_i = .80197872 < U(2) = .828$$

F_5: .8747516641

$$\sum_{i=17}^{17} x_i = .8747516641 < U(1) = 1$$

Finally, an example will be given of a data set where rearrangement in theorem 1 is actually necessary. If the artificial intelligence functions of the example are considered separately and their utilizations x_i are calculated, the following assignment to three processors $\{F_i\}$ is obtained.

F_1: .05568<.06007<.100425<.119088876 <.1443262

$$\sum_{i=1}^{5} x_i = .479590076 < U(5) = .743$$

F_2: .2814787< .3227064 (3)

$$\sum_{i=6}^{7} x_i = .6041851 < U(2) = .828$$

F_3: .4689868

$$\sum_{i=8}^{8} x_i = .4689868 < U(1)$$

The AI functions considered here are : Ranking of Candidate Reactions, Reaction Selection, Force Resources Evaluation, Target Evaluation, Force Data, Result Evaluation, Candidate Reaction Evaluation, Target Data. The utilizations obtained firstly by truncating those of the system (3) to two decimals, except in the case of .119088876 which is rounded to .12 and secondly by changing the utilization on F_3 to .36 instead of .4689868 can be shown to correspond to a task set that can be rearranged on a smaller number of processors. The original assignment as defined in theorem 1 gives:

F_1: .05<.06<.1<.12<.14

$$\sum_{i=1}^{5} x_i = .47 < U(5) = .743$$

F_2: .28 < .32

$$\sum_{i=6}^{7} x_i = .6 < .828$$

F_3: .36

$$\sum_{i=8}^{8} x_i = .36 < 1$$

These tasks may be rearranged on F_2 and F_3 thus:

F_2: .05<.1<.28<.32

$$\sum_{i=1}^{4} x_i = .75 < U(4) = .756$$

F_3: .06<.12<.14<.36

$$\sum_{i=5}^{8} x_i = .68 < U(4) = .756$$

Thus, the eight tasks have been rearranged on two processors.

CONCLUSIONS

In this paper, algorithms are put forward to describe the hardkill and softkill reactions that can be chosen by the knowledge-based TEWA of a manoeuverable ship which is attacked by anti-ship missiles. A methodology is also described for

designing real-time knowledge bases on transputer networks. The description includes a discussion of some of the different viewpoints and difficulties of concurrent computing using transputers. Finally, some theoretical results are obtained for multiprocessor scheduling of artificial intelligence functions on RISC processors and these results are illustrated using some simple shipboard anti-air warfare AI functions.

References

1. Carling, R.L., "A Knowledge-Base System for the Threat Evaluation and Weapon Assignment Process", *Naval Engineers Journal*, vol. 105, no. 1, pp. 31-41, January 1993.

2. Carling, R., "An Artificial Intelligence Approach to the Threat Evaluation and Weapon Assignment Problem", *Maritime Engineering Journal*, pp. 12-16, January 1993.

3. Thomson CSF Systems Canada, "TEWA Simulator Design Report", DSS Contract No.: W7701-8-4770/01-XSK, TCSC Report No.: C0048C-01, July 1991.

4. Thomson CSF Systems Canada, "Software Design Document for the Modelled Entities of the Anti-Air Warfare Simulator", DSS Contract No.: W7701-0-3202/01-XSK, TCSC Report No.: G0456C-06, November 1992.

5. Thomson CSF Systems Canada, "TEWA Resource Loading Study, Ship AAW System Model Specification", DSS Contract No.: W7701-9-4137/01-XSK, January 1991.

6. Thomson CSF Systems Canada, "TEWA Resource Loading Study, Ship Convoy Model Specification", DSS Contract No.: W7701-9-4137/01-XSK, TCSC File No.: G0194C, August 1991.

7. Thomson CSF Systems Canada, "TEWA Resource Loading Study, Simscript Model Design Document", DSS Contract No.: W7701-9-4137/01-XSK, TCSC File No.: G0194C, January 1993.

8. Green, D., Liscano, R., and Wein, M.,"Real-Time Control of an Autonomous Mobile Robot using the Harmony Operating System", *Proceedings of the IEEE International Symposium on Intelligent Control 1989*, Albany, New York, pp. 374-378, 25-26 September 1989.

9. Liscano, R., Fayek, R.E., Karam, G.M., "A Blackboard, Activity-Based Control Architecture for a Mobile Platform", NRC Report No. 33196.

10. Chandrasekaran, B., "Generic Tasks in Knowledge Based Reasoning: High-Level Building Blocks for Expert System Design", *IEEE Expert*, pp. 23-30, Fall 1986.

11. de Carlini, U., and Villano,U., "Transputers and Parallel Architectures: message-passing distributed systems", Ellis Horwood Limited, Chichester, England, 1991.

12. INMOS, "The Transputer Applications Notebook: Systems and Performance", first edition, Bristol, England, June 1989.

13. INMOS, "The Transputer Applications Notebook: Architecture and Software", first edition, Bristol, England, May 1989.

14. Finn, A., Griffin, M., and McClurg,C., "Modelling and Simulation of an i860-Based Multiprocessor", *24th Annual Simulation Symposium*, New Orleans, LA, 1991.

15. Lina, J-M., Scott, C.K., Goulard, B., and Mayrand M., "Advanced Simulation of CANDU reactors at low cost with parallel processing", *17th CNS Simulation Symposium*, Kingston, 1992.

16. Stankovic, J.A., and Ramamritham, K., "Hard Real-Time Systems", IEEE Computer Society Press, Los Alamitos, CA, pp. 174-189, 1988.

17. Sha, L., and Goodenough, J.B., "Real-Time Scheduling Theory and Ada", *IEEE Computer*, pp. 53-62, April 1990.

18. Sprunt, B., Sha, L., and Lehoczky, J.P., "Aperiodic Task Scheduling for Hard-Real-Time Systems", *J. of Real-Time Systems*, Vol. 1, pp. 27-60, 1989.

19. Encore 93 Series Technical Summary, Encore Computer Corp., Fort Lauderdale, FL, 1991.

20. "The Handbook of Real-Time Systems Analysis: Based on the Principles of Rate Monotonic Analysis", *Carnegie Mellon University, Software Engineering Institute*, July 1992.

21. Rajkumar, R., Sha, L., and Lehoczky, J.P., "Real-Time Synchronization Protocols for Multiprocessors", *Proc. IEEE Real-Time Systems Symp.*, Los Alamitos, CA, pp. 259-269, 1988.