A Concurrent Distributed System for Aircraft Tactical Decision Generation

John W. McManus
NASA Langley Research Center
Hampton, Virginia 23665-5225

ABSTRACT

A research program investigating the use of artificial intelligence (AI) techniques to aid in the development of a Tactical Decision Generator (TDG) for Within Visual Range (WVR) air combat engagements is discussed. The application of AI programming and problem solving methods in the development and implementation of a concurrent version of the Computerized Logic For Air-to-Air Warfare Simulations (CLAWS) program, a second generation TDG, is presented. Concurrent computing environments and programming approaches are discussed and the design and performance of a prototype concurrent TDG system are presented.

INTRODUCTION

The increased capabilities of modern weapons and sensor systems and the expanded capabilities and flight envelopes of high-performance aircraft have changed the requirements of air combat simulation systems. A modern and realistic air combat simulation that can be used to evaluate the current and future air combat environments must have the ability to model superagile aircraft as well as new weapons systems, aircraft subsystems such as sensors or propulsion systems, modifications to existing aircraft control systems, and changes to the aircraft's structural configuration. In support of the study of superagile aircraft at Langley Research Center (LaRC), a Tactical Guidance Research and Evaluation System (TGRES, pronounced "tigress") is being developed.[1,2,3]

The TGRES system[1,2,3], shown in figure 1, is designed to allow researchers to develop and evaluate systems in a tactical environment. While TGRES is aimed specifically at the development and evaluation of maneuvering strategies and advanced guidance/control systems for superagile aircraft, the modular design of TGRES will make it easily adaptable to the analysis of other aircraft systems. The three main components of TGRES are a Tactical Decision Generator (TDG), the Tactical Maneuver Simulator (TMS), and the Differential Maneuvering Simulator (DMS). Both the TMS and the DMS use a TDG as an automated opponent. The TMS and the DMS are described in greater detail in [1,3]. This paper describes the design, implementation, and efficiency of a prototype concurrent TDG.

TGRES COMPONENTS

The TMS[1,3] provides a high-fidelity batch air combat simulation environment for the development and testing of various guidance and control strategies. The researcher defines the initial conditions of the engagement and the TMS then executes the trajectories and attitudes of the aircraft using simple trajectory commands or through a tactical guidance system. The main elements of the TMS are a high-fidelity, nonlinear six degree-of-freedom (dof) rigid-body dynamic aircraft model, a TDG, and a user interface.[3]

The DMS consists of two 40' diameter domes and one 20' diameter dome located at Langley Research Center. The facility is intended for the real-time simulation of air combat engagements between piloted aircraft. By using a TDG to control one of the airplanes, it is possible to test a TDG against a human opponent. This feature allows the guidance logic to be evaluated against an unpredictable and adaptive human opponent.

CONCURRENT CLAWS

A concurrent version of the Computerized Logic For Air-to-Air Warfare Simulations (CLAWS) software, Cube Claws, was developed as a distributed blackboard system in C on a Intel Hyper Cube. CUBE CLAWS simulates one-on-one air combat engagements. A blackboard system consists of a set of specialized knowledge sources, a centralized blackboard data structure, and a control strategy used to activate the knowledge sources. The blackboard is a global data structure, often partitioned in a hierarchical manner, used to represent the problem domain[4]. The blackboard is also used to allow interknowledge source communication and acts as a global shared memory visible to all of the knowledge sources. This design provides for opportunistic problem solving and allows a knowledge source to contribute towards the solution of the current problem without knowing which of the other knowledge sources will use the information.

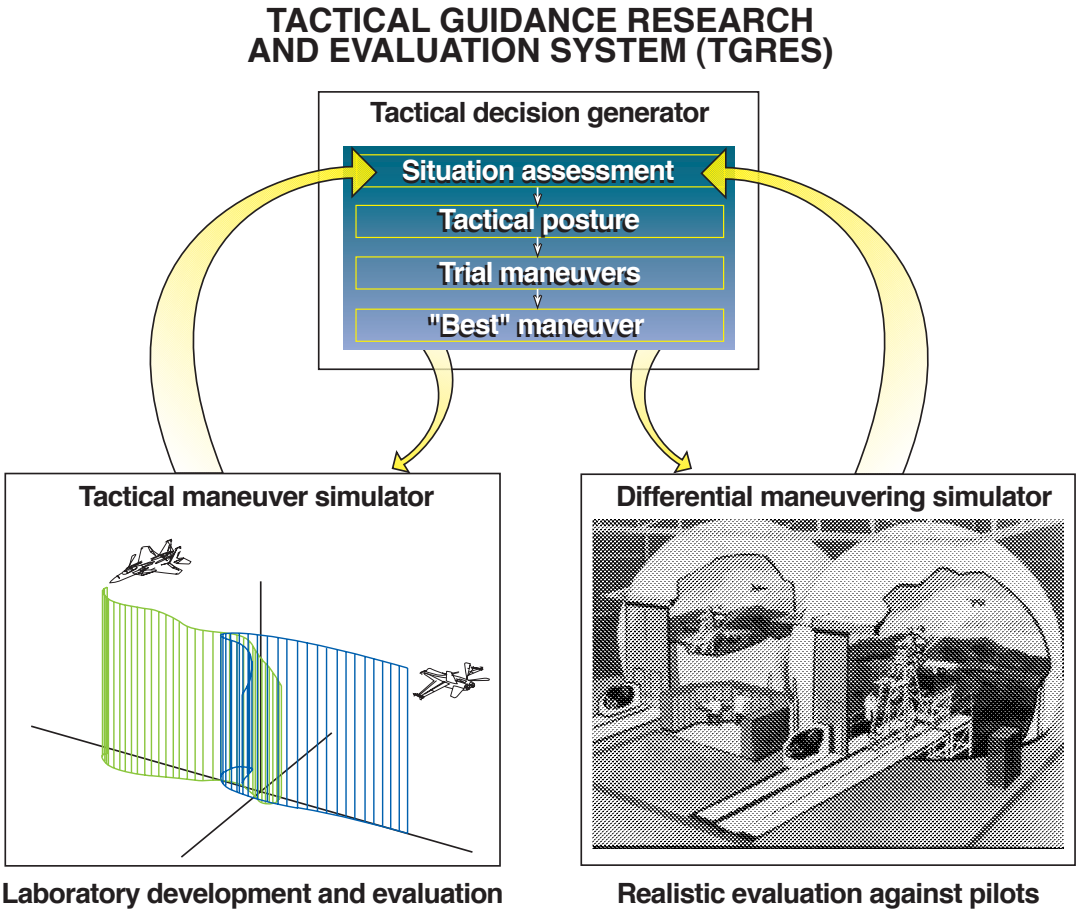

Figure 1. TGRES

Cube CLAWS has been designed with separate subroutines for the aircraft simulation and the TDG knowledge sources. The separation of the aircraft simulation and decision logic components and the use of highly specialized knowledge sources allows each module/knowledge source to be designed and implemented on separate processors on the hypercube. Each knowledge source can be developed and tested independently before it is incorporated into Cube CLAWS.

The independence of the knowledge sources also increases the efficiency of Cube CLAWS by allowing knowledge sources to be distributed across the hypercube. The knowledge sources communicate with the blackboard using a message passing system.

The Cube CLAWS is a distributed blackboard system designed to execute on a 16 processor Intel IPSC HyperCube. Cube CLAWS consists of a set of knowledge sources for both aircraft in the simulation. The set includes: a main knowledge source that contains the blackboard support software and aircraft model, a relative geometry knowledge source, a situation assessment knowledge source, and a maneuver evaluation knowledge source used to evaluate a set of eight prospective aircraft maneuvers. The control structure used for activating knowledge sources is message driven and is embedded in the knowledge sources. The blackboard elements are passed as messages to and from the modules, and read/write synchronization is used to ensure blackboard consistency.

The Cube CLAWS software consists of six basic files: local.h, host.c, main1.c, relative1.c, sit1.c, and eval1.c. The file local.h contains the global data definitions and the declarations for the blackboard structures. Host.c is the host program used to load the other Cube CLAWS software onto the assigned processors and initialize the aircraft parameters for both aircraft in the simulation. The host program then initiates the blackboard system by passing the initial aircraft states to the two instances of the main knowledge source. The host

program then goes into a loop to receive messages containing the updated aircraft states and writes them to the console. Main1.c contains the code for an instance of the main knowledge source; relative1.c contains the code for an instance of the relative geometry knowledge source; sit1.c contains the code for an instance of the situation assessment knowledge source; and eval1.c contains the code for an instance of the maneuver evaluation knowledge source.

The main knowledge source contains the aircraft modeling software and the required blackboard control elements. The main knowledge source is in effect the "controlling" module for both aircraft in the engagement. The main knowledge source receives the initial aircraft state for the aircraft it will be controlling and then initializes the variables used for interknowledge source communication. The main knowledge source then swaps aircraft state information with each other main knowledge source so that at the start of each time step all main knowledge sources know the position and appropriate state variables for all other aircraft in the engagement. The aircraft controlled by the main knowledge source is called "aggressor" and the other aircraft is called "target."

The relative geometry knowledge source is activated to compute the relative geometry between the two aircraft and then the situation assessment knowledge source is activated to determine the aircraft's current mode of operation. The maneuver evaluation knowledge source (Eval) then evaluates prospective maneuvers and returns the control commands for the highest rated maneuver to the main knowledge source. The maneuver is executed and the next cycle of the engagement begins.

The relative geometry knowledge source uses the current state of the aggressor and target aircraft stored on the blackboard. The line-of-sight (LOS) angle between the aggressor and the target, the closing rate, the distance, and the angle-off (the angle between the LOS vector and the x body axis) are computed to be returned to the blackboard. The weapons solutions are then computed and if a gun or missile lock is achieved, the weapons systems are activated and the weapons lock times are incremented. The updated blackboard values are returned to the main knowledge source at the completion of the knowledge source execution.

The situation assessment knowledge source uses both the data returned by the relative geometry knowledge source and other aggressor aircraft state data to compute the current tactical situation and update the aggressors mode of operation. A set of eight tactical evaluation metrics are used to define the situation space. The situation assessment knowledge source uses a fuzzy logic based scoring scheme to evaluate the metrics and map the current situation into one of the five modes of operation shown in table 1. The situation assessment knowledge source also computes the target's current mode of operation. The mode of operation is used to select a set of scoring weights that are used to generate a numeric "score" for the current maneuver. The score of a maneuver represents the computed tactical worth of the position being evaluated. The updated blackboard data elements are returned to the main knowledge source at completion of the knowledge source execution.

Modes of Operation
Aggressive
Evasive
Missile Evasion
Ground/Stall Evasion
Evading opponent's "lock-on"
Defensive
Neutral
Bugout

Table 1.

The evaluation knowledge source is used to evaluate the tactical value of candidate maneuvers. The knowledge source uses the current state of the aggressor and target aircraft from the blackboard. The Eval routine first predicts the future position of the opponent based on his last known position, flight path angle, speed, and heading. Eval then generates a set of eight candidate maneuvers based on the current mode of operation and generates the new position for the aggressor aircraft. These new positions and the projected position of the opponent are placed in blackboard data elements and the relative geometry knowledge source is activated to compute the relative geometry between the candidate positions and the predicted position of the opponent. The results are placed in blackboard elements and the situation assessment knowledge source is activated to compute the mode of operation for each of the maneuvers and a tactical score for each position. The results are then placed in blackboard elements. Eval selects the highest scoring maneuver and the required control commands are placed on the blackboard to be used by the main knowledge source.

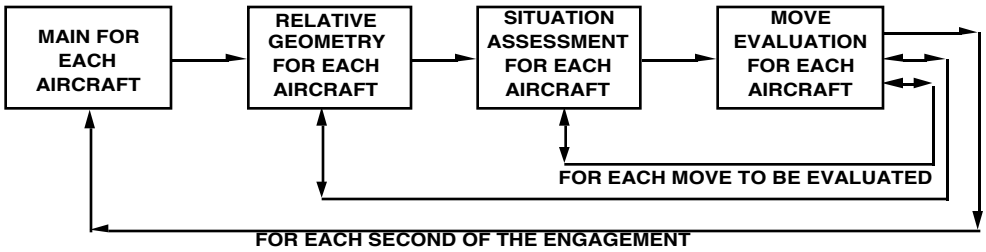

Figure 1. CLAWS Schematic

Cube CLAWS Software Structure

Figure 1 shows a schematic of the "optimal serial" version of the Cube CLAWS program. The optimal serial version is a highly optimized serial version of the CLAWS software. The main knowledge source is called for both aircraft, followed by a call to compute the relative geometry between the two aircraft and a call to perform the situation assessment. The move evaluation subroutine is then called and it calls an instance of the relative geometry and the situation assessment once for each prospective maneuver to be evaluated. This cycle is executed once for each second in the simulated engagement. The Cube CLAWS exploits the natural parallelism of the engagement by creating separate parallel execution paths for both aircraft in the engagement. The main knowledge source for each of the aircraft synchronizes at the start of each time step in the engagement to swap aircraft state data and then proceeds down parallel execution paths.

The evaluation of trial maneuvers is also done in parallel. Multiple versions of the situation assessment and relative geometry knowledge sources are loaded onto processors of the cube and are used to evaluate the candidate maneuvers. The Eval module generates the prospective maneuvers and then sends one maneuver to each available relative geometry knowledge source. When all of the maneuvers have been distributed and processed, the results are placed on the blackboard and the maneuvers are distributed to the available situation assessment knowledge sources. The results are then evaluated and the resulting set of control commands is placed on the blackboard for the main knowledge source to execute.

It is important to note that although multiple versions of the relative geometry modules and the situation assessment modules are being executed in parallel, there is still an inherent serialization between the relative geometry and the situation assessment modules. The relative geometry must be computed for a maneuver before the situation assessment module can begin execution. Figure 2 is a schematic of the current Cube CLAWS software configuration.

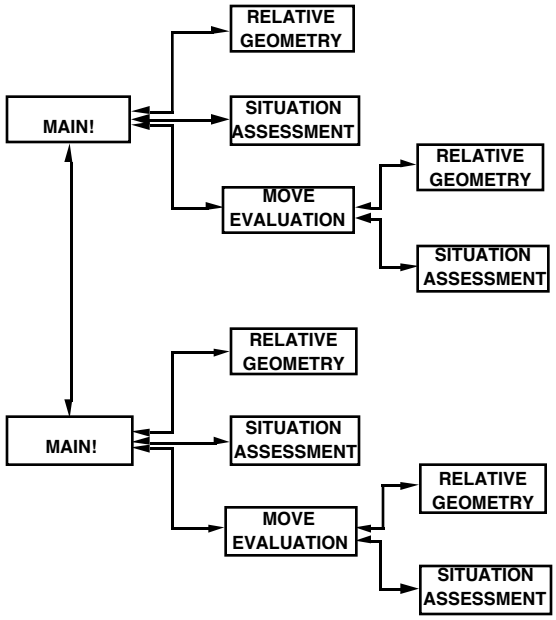

Figure 2. Cube CLAWS Schematic

Evaluation of Cube CLAWS Performance

A total of six test cases were used to evaluate the performance of Cube CLAWS. The baseline case was the optimal serial version of Cube CLAWS run on a single processor. The Cube CLAWS software was then run in the configurations described in table 1, with the table entries representing the hypercube processor executing the software. Configuration one (C1) loaded all of the processor software for both aircraft on a single processor. Configuration two (C2) loaded the processor software for aircraft one (A1) on processor 0 and the processor software for aircraft two (A2) on processor 1, creating parallel execution paths for the two aircraft. Configurations three through five (C3, C4, C5) loaded the processor software as described for configuration two with the addition of loading multiple versions of the situation assessment and relative geometry software.

Version	Aircraft	Main	Relative Geometry	Situation Assessment	Evaluation
C1	A1	0	0	0	0
	A2	0	0	0	0
C2	A1	0	0	0	0
	A2	1	1	1	1
C3	A1	0	0,2,3	0,2,3	0
	A2	1	1,8,9	1,8,9	1
C4	A1	0	0,2,3,4,5	0,2,3,4,5	0
	A2	1	1,8,9,10,11	1,8,9,10,11	1
C5	A1	0	0,2,3,4,5,6,7,8,9	0,2,3,4,5,6,7,8,9	0
	A2	1	1,8,9,10,11,12,13,14,15	1,8,9,10,11,12,13,14,15	1

Table 2. Software Test Configuration Processor Assignments

Speedup and efficiency computations were performed for the evaluation module and per second of the engagement. Speedup is a measure of how much faster a problem P is solved using N processors than when solved serially. The efficiency of a parallel algorithm is defined to be the speedup divided by the number of processors used. Due to the size of the problem it was not possible to run the eight processor versions of Cube CLAWS and evaluate all eight potential maneuvers without having some processors executing more than one task (processor overlap). In these cases the speedup without processor overlap was computed using the average execution speed of the nonoverlapping processors.

Figure 3 plots the expected speedup per second of simulated engagement (in the case of processor overlap) and the speedup actually achieved. Configuration one is used as the base case for these speedup calculations. These calculations measure the effect of splitting the task into separate execution paths for each aircraft. Note that the optimal serial implementation took approximately 23 percent less time to execute than configuration one. The speedup gained by splitting into separate paths for each aircraft is found by comparing the configuration one and the configuration two data. In this case the speedup achieved is almost linear, approximately 97 percent. There is a distinct leveling off in speedup between configuration four and configuration five. Configuration five again achieves much better speedup when no process overlap occurs.

Figure 4 shows that processor efficiency decreases as additional processors are added. Much of the loss of speedup is due to the serial nature of the main knowledge source and the need for all main knowledge sources to synchronize at the start of each iteration. It is important to realize that the additional processors are only being assigned work by the evaluation knowledge source and do not speed up the execution of the aircraft models. The additional processors are idle except when evaluating trial maneuvers.

The computed speedups for the maneuver evaluation knowledge source (without processor overlap) are shown in figure 5. These calculations measure the effect of adding additional processors for the relative geometry and situation assessment knowledge sources on the execution of the Eval knowledge source. The speedup is close to linear for all cases except configuration five, the eight processor case. This data shows that adding additional processors has a large effect on the execution time of the maneuver evaluation subroutines.

As shown in figure 6 the processor efficiency is greater than 65 percent in all configurations tested. Configuration 5 shows significant improvement in both speedup and processor efficiency. The computed efficiency for

configuration five has increased to approximately 67.5 percent.

It is important to note that the sequential relationship between the situation assessment knowledge source and the relative geometry knowledge source reduces the benefit of adding additional processors in the current software configuration and increases the number of messages that must be passed. The cost of passing these messages, both in time and operating system overhead, can be very large. The sequential relationship implies that the system may perform better if the two separate knowledge sources are combined to form a single knowledge source.

Concluding Remarks

The speedup and efficiency data for the evaluation knowledge source are very promising and the overall speedup and efficiency data for the main knowledge source show that there is a clear advantage to splitting the problem into parallel execution paths for each aircraft. The data also highlighted some inefficiencies that may be corrected by redesigning parts of the system. Much of the loss of efficiency in the evaluation subroutine can be attributed to the serial link between the relative geometry knowledge source and the situation assessment knowledge source; and to the ratio of execution time for these relatively small knowledge sources to the time they required to send and receive messages. These knowledge sources can be combined into a single specialized knowledge source. This will reduce the evaluation knowledge sources message passing time, cut the number of messages required, and remove the synchronization requirement between the two separate relative geometry and situation assessment knowledge sources.

The Cube CLAWS has provided a useful testbed to evaluate the development of a distributed blackboard system. The project has shown that the complexity of developing specialized software on a distributed, message-passing architecture such as the Hypercube is not overwhelming and that reasonable speedups and processor efficiency can be achieved by a distributed blackboard system. The project has also highlighted some of the costs of using a distributed approach to designing a blackboard system. Message passing costs, synchronization costs, and the cost of having multiple processes executing on a processor must be recognized during the system design phase so that their effect on the systems performance can be minimalized.

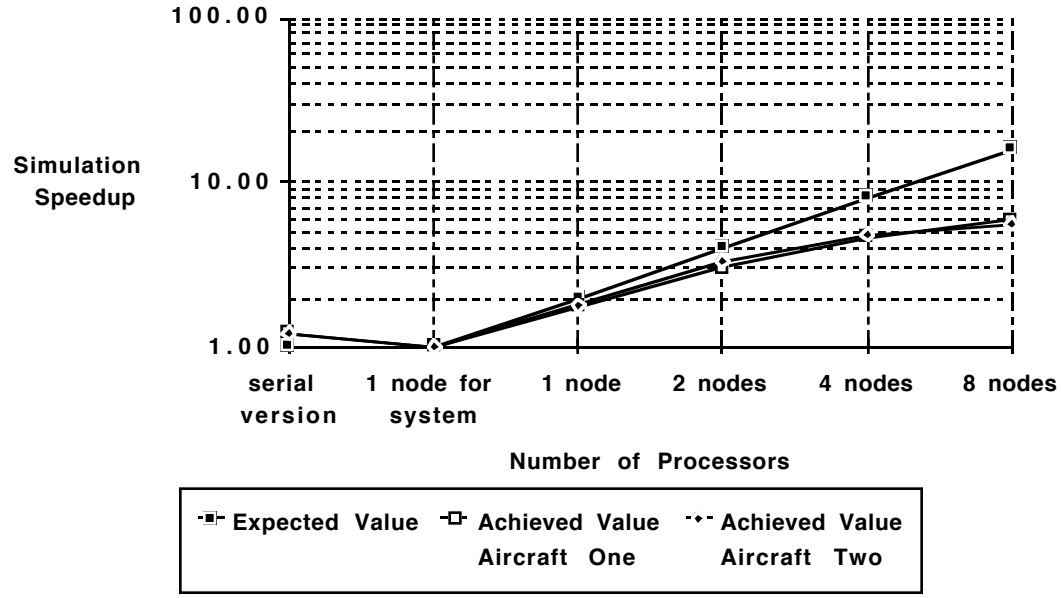

Figure 3. Simulation Speedup

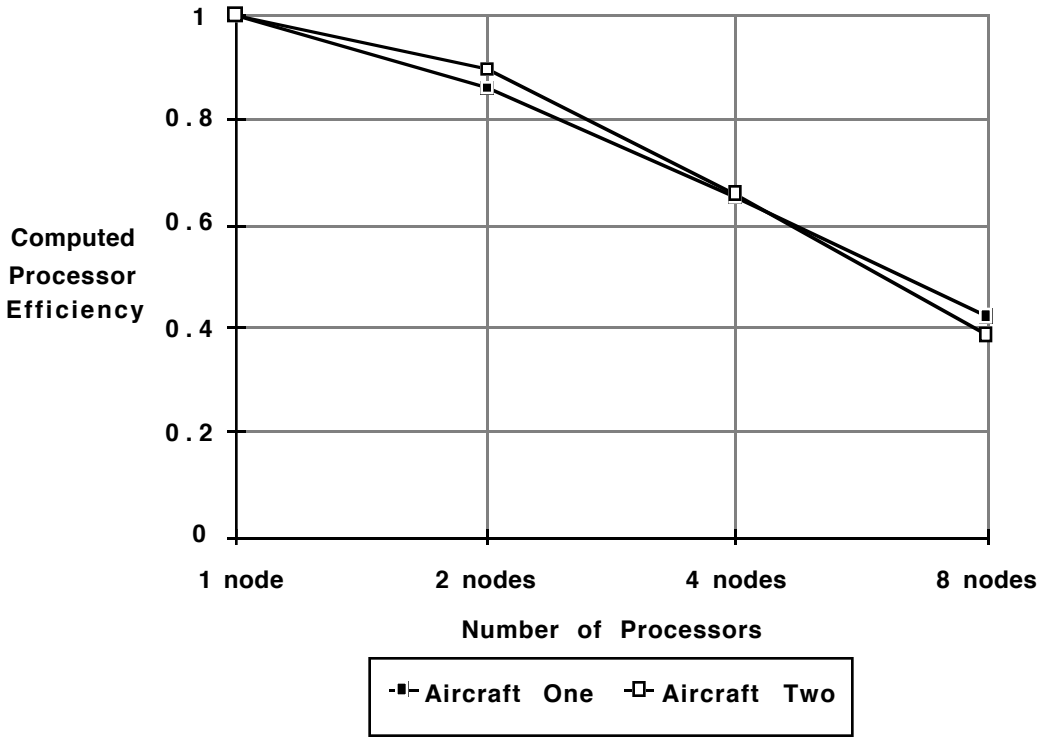

Figure 4. Processor Efficiency for Simulation.

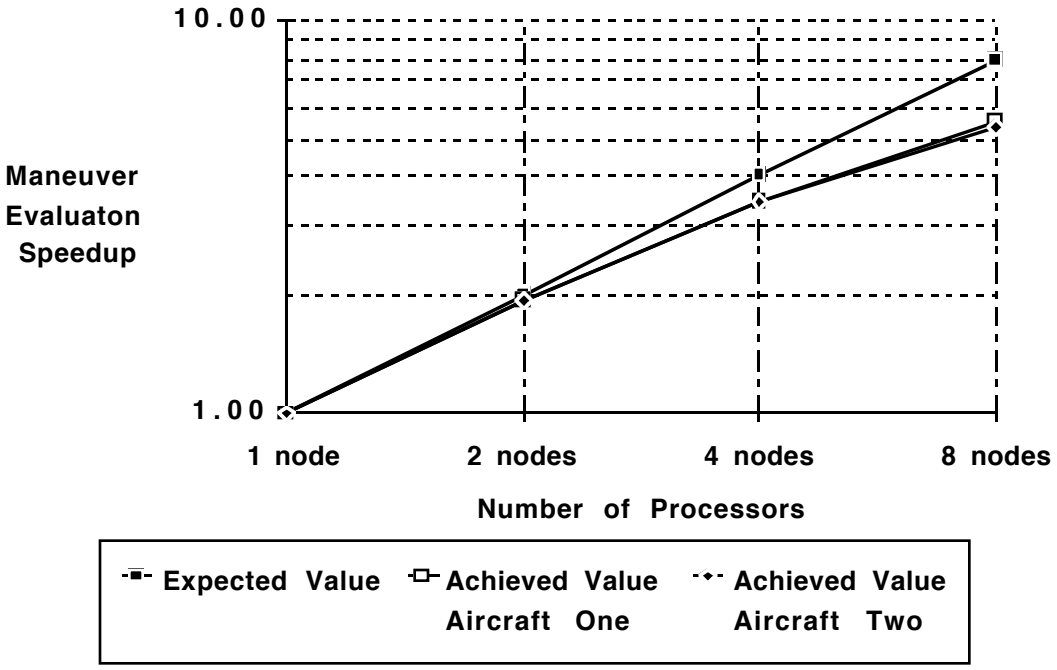

Figure 5. Maneuver Evaluation Speedup

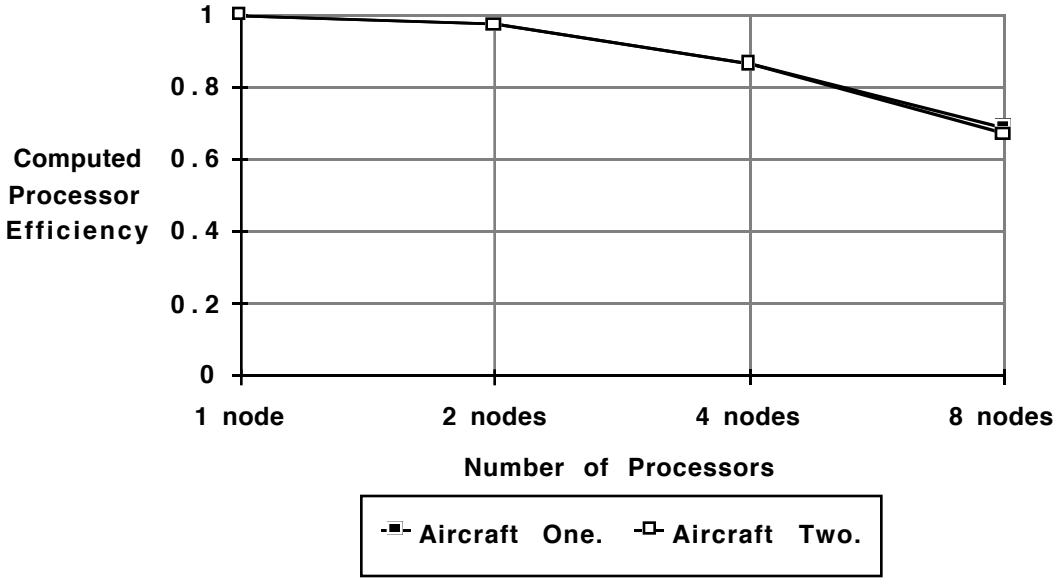

Figure 6. Processor Efficiency for Maneuver Evaluation Subroutine

REFERENCES

1. Goodrich, Kenneth H; McManus John W. : "*Development of A Tactical Guidance Research and Evaluation System (TGRES).*" AIAA Guidance, Navigation and Control Conference. AIAA Paper #89-3312, August 1989.

2. McManus, John W.; Goodrich, Kenneth H. : "*Application of Artificial Intelligence (AI) Programming Techniques to Tactical Guidance for Fighter Aircraft.*" AIAA Guidance, Navigation and Control Conference. AIAA Paper #89-3525, August 1989.

3. Goodrich, Kenneth H; McManus John W. : "*An Integrated Environment For Tactical Guidance Research and Evaluation.*" AIAA Flight Test Conference. AIAA Paper #90-1287, May 1990.

4. Ni, H. Penny : "BlackBoard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures." *AI Magazine*, 7 (3), 1986, pp. 38 - 53.