1-1-1994

Intervisibility Heuristics For Computer Generated Forces

Sumeet Rajput

INSTITUTE FOR SIMULATION AND TRAINING

Contract Number N61339-92-C-0045
September 26, 1994

Intervisibility Heuristics for Computer Generated Forces

Institute for Simulation and Training
3280 Progress Drive
Orlando FL 32826

University of Central Florida
Division of Sponsored Research

IST-TR-94-22

INSTITUTE FOR SIMULATION AND TRAINING

Intervisibility Heuristics for Computer Generated Forces

Contract Number N61339-92-C-0045
September 26, 1994

IST-TR-94-22

Prepared by:

Sumeet Rajput
Michael A. Craft
Larry J. Breneman
Mikel D. Petty
Clark R. Karr
Thad P. Holly
Jason J. Ng

Reviewed by:

Michael A. Companion

Institute for Simulation and Training • 3280 Progress Drive • Orlando, Florida 32826

University of Central Florida • Division of Sponsored Research

Intervisibility Heuristics for Computer Generated Forces

IST-TR-94-22

Contract N61339-92-C-0045
September 26, 1994

Sumeet Rajput, Michael A. Craft, Larry J. Breneman, Mikel D. Petty, Clark R. Karr
Thad P. Holly and Jason J. Ng

Table of Contents

List of figures

List of tables

1. Executive summary

A Computer Generated Forces (CGF) system must both generate behavior and perform physical modeling for the entities it is controlling. The process of determining when entities "see" one another is an essential component of modeling behavior and combines both behavior and physical modeling. Many factors (behavioral and physical) are involved in one entity "seeing" another; e.g. attention, angle of view, weather, obscurants, time of day, and sensor system. These factors are modeled in each CGF system's "sighting model" which may differ among CGF systems. However, all sighting models are based on the existence or non-existence of a unblocked straight line between the entities, the "line of sight". This line of sight represents the path of light between entities. Every "sighting determination" rests on an "intervisibility determination" which determines whether the line of sight between two entities is blocked by the surface of the terrain, terrain features, or intervening entities.

In order to generate behavior that is responsive to changes in battlefield situations, a CGF system must perform intervisibility determinations between each entity it is controlling and each hostile entity in the simulation at frequent intervals, typically once per second. Thus, in a scenario of even moderate size a great many intervisibility determinations must be made. Moreover, each entity-to-entity intervisibility determination can require considerable computational expense as the line segment that represents the line of sight is tested for intersection with the terrain, feature, and entity polygons that lie along it. It is very possible for a CGF system to spend as much as 50% of its computational power performing this one operation.

In earlier work, researchers at the Institute for Simulation and Training (IST) developed three new algorithms for the intervisibility determination process. The best of those algorithms was 25% faster than the best previously available. Unfortunately, the new algorithms required specific terrain database formats to operate, a fact that limited their applicability to the IST CGF Testbed.

In the research described in this report, IST has taken a different approach. Rather than trying to *speed each individual intervisibility determination*, the goal of this work was to *reduce the number of intervisibility determinations* done by a CGF system. This was accomplished through a set of intervisibility heuristics. A heuristic is a "rule of thumb" or a "guess" that is right most of the time. Intervisibility heuristics are algorithmic "rules of thumb" that determine when an intervisibility determination can be skipped or delayed by the CGF system. Note that this approach is independent of terrain database format.

Of course, simply reducing the number of intervisibility determinations is of questionable value if the realism of the CGF behavior is degraded by the heuristic. The intervisibility heuristics described in this report were designed to delay the first sighting of a hostile entity by as little time as possible and thereby have minimal impact on the CGF behavior.

Seven intervisibility heuristics were designed. The heuristics, and their basic ideas, are:

1. **Symmetry**; the result of a determination from entity A to entity B will also apply from B to A until one of them moves.

2. **History**; if two entities have (or have not) had a line of sight for some time, they are likely to continue to have (or not have) a line of sight in the immediate future.

3. **Sighter**; certain characteristics of the sighter, such as its firepower status or proximity to hostile entities, make it more or less likely to need intervisibility determinations from it.
4. **Target**; certain characteristics of the target, such as its damage status or movement direction relative to hostile entities make it more or less likely to need intervisibility determinations to it.

The History, Sighter, and Target heuristics were also implemented in fine-grain versions which are based on the same ideas but use a finer time granularity when delaying intervisibility determinations.

All seven of these heuristics were implemented within the IST CGF Testbed. To evaluate their performance, they were tested individually in three scenarios of different military types (Meeting Engagement, Delay, and Assault). Data was gathered for each heuristic as to how many intervisibility determinations it saved and by how much it delayed the sighting of each hostile entity.

All of the heuristics performed well at reducing intervisibility determinations with the savings ranging from 10% to 50%. Even taking the overhead of computing the heuristic into account, significant reductions in computational load associated with intervisibility processing were achieved.

These computational savings had negligible negative effect on the behavior of the CGF entities. The heuristics caused an average delay in sightings by the CGF entities of 0.3 to 0.5 seconds, i.e. of less than one second.

These results can be of great importance to CGF systems. By using one of these heuristics, the computational load of intervisibility determination can be greatly reduced, thereby freeing computational capacity that can be applied to generating more sophisticated behavior, performing more realistic physical modeling, or simply controlling more entities on a given system. Because these heuristics are completely independent of the terrain database format, they can be applied to any CGF system.

2. Introduction

2.1. Purpose

This technical report is a deliverable under STRICOM contract N61339-92-C-0045, "Intelligent Autonomous Behavior by Semi-Automated Forces in Distributed Interactive Simulation". It comprises both CDRL A004 "Line of Sight Technical Report", which is the main body of the report, and CDRL A003 "Line of Sight Test Data", which is Appendix A.7.

Its purpose is to report the methodology and results of experimental software development conducted at the Institute for Simulation and Training (IST) under that contract. The goal of the software development was to produce heuristics that would reduce the overall computational expense of intervisibility determination in Computer Generated Forces (CGF) systems without affecting the realism of the autonomous behavior produced by those systems.

2.2. Structure of this document

The remainder of this section contains background information on Distributed Interactive Simulation, Computer Generated Forces, the IST CGF Testbed, and IST's research results in the CGF area. Readers familiar with these topics may safely skip those subsections without loss of continuity.

Sections 3 and 4 introduce concepts of intervisibility in Distributed Interactive Simulation (DIS) and CGF systems and explain how intervisibility is calculated in the IST CGF Testbed. Section 5 describes the intervisibility heuristics developed by IST and how they were implemented. Section 6 presents the experimental evaluation of the implemented heuristics and the results of that evaluation. Section 7 states the conclusions drawn from those results.

Section 8 lists the references cited in the report. Finally, the Appendices section contains a variety of supplemental information about the heuristics and the experiment that may be of interest to some readers.

2.3. Background

2.3.1. Distributed Interactive Simulation

Distributed Interactive Simulation is an architecture for building large-scale simulation models from a set of independent simulator nodes (DIS[1993]). The simulator nodes are linked by a network and communicate via a common network protocol. (The term DIS is also sometimes used to designate a particular network protocol standard; in this document "DIS" refers to the simulation architecture; the DIS protocol standard will be so identified.) In DIS, the simulator nodes each independently simulates the activities of one or more entities in the simulated system and report their attributes and actions of interest to other simulator nodes via the network. The simulated entities coexist in a common environment (for example, a terrain database) and interact by exchanging network packets (Loper et. al.[1991]). Finally, an important characteristic of DIS simulations is that they are real-time; events in the simulation occur at the same rate as their real-world counterparts.

The DARPA/US Army SIMNET system is a DIS-type system. SIMNET is used to train tank and vehicle crews in cooperative and team tactics. Thorpe[1987] and Pope[1989] are good examples of the extensive SIMNET literature. In SIMNET, the simulator nodes, or

simulators, typically represent single vehicles, such as tanks or infantry fighting vehicles. SIMNET simulators are complex devices consisting of a simulation computer, a computer image generator, and a physical re-creation of the vehicle interior; they are manned by three or four human trainees. During the execution of a scenario, each simulator's simulation computer continuously tracks the position of the vehicle on a terrain database common to all vehicles in the scenario. The trainees maneuver their vehicle over the terrain and interact with (for example, fire their weapons at) other vehicles. The simulators are linked by an Ethernet network, which carries the packets needed to mediate inter-vehicle interaction.

By exchanging these packets, actions taken by one simulator are made known to other simulators in real-time. Each vehicle broadcasts appearance packets, which are used by other vehicles to generate visual images, and fire and impact packets, which are used to signal and adjudicate weapons firing.

A DIS system, of which SIMNET is the archetype, depends on two points of commonality among the networked simulators. The first point is the shared "playing field", or simulated environment. In order that events and actions could be consistent among simulators, the simulated environments must be either identical or isomorphic. For example, if two tank simulators share terrain databases which are identical except that a bridge present in the first is omitted in the second, when the first tank traverses the bridge it will appear to the second to be floating on air or water.

The second required point of commonality is the network protocol. A DIS protocol specifies the various types and formats of network packets which the simulator nodes will exchange to support the simulation model. Additionally, the protocol defines the precise circumstances under which each packet type should be sent by a simulator node, and the interpretation that should be performed when each packet type is received.

2.3.2. Computer Generated Forces

In the case of SIMNET and its currently envisioned DIS successors, the system is designed to provide a simulated battlefield which is used for training military personnel. In such a battlefield, the trainees need an opposing force against which to train. There are at least three ways to provide the simulated opponent.

In the first method, two groups of trainees in simulators may oppose each other. For example, it is possible to configure SIMNET simulators during startup so that the computer image generators in each force's simulators display their own force's vehicles as US vehicles (M1 Abrams and M2 Bradleys) and the opposing force's vehicles as enemy vehicles (T-72s and BMPs). Thus both sides see themselves as US forces and their opponents as the enemy. This method is often used, and the soldiers enjoy the competitive aspects of the arrangement, but it has several disadvantages. It increases the number of expensive simulators needed at a training site and requires that to train any given military unit a second unit be available to provide the opposition. Finally, the trainees are faced with opponents who, despite the appearance of their vehicles in the video screens, use U.S. Army tactical doctrine because U.S. Army soldiers are controlling the vehicles. It would be preferable to provide the trainees with opponents who use the tactical doctrine of the actual or anticipated enemy.

A second method is to use human instructors who are trained to behave in a way that mimics the desired enemy doctrine. Doing so does not reduce the need for simulators and is expensive in manpower costs because large numbers of trained instructors may be required. Furthermore, the instructors must be retrained for each new enemy's doctrine. This method is seldom used.

The third technique is to use a computer system that generates and controls multiple simulation entities using software and possibly a human operator. This type of system is known as a semi-automated force (SAF or SAFOR) or a computer generated force (CGF). The following sections will describe, in general terms, the features and structure of CGF systems.

2.3.2.1. Characteristics of CGF systems

Certain characteristics are common to all existing CGF systems. Some of the most important of those characteristics are listed below and each will be described in turn.

1. Network connection and protocol
2. Battlefield environment simulation
3. Support for multiple entities
4. Autonomous behavior generation
5. Operator control of behavior
6. Representation of the military organizational hierarchy

2.3.2.1.1. Network connection and protocol

Because DIS-type simulations are networked, a CGF system needs both a physical connection to the network and the appropriate software to send and receive network packets. Furthermore, the system must conform to the network protocol that has been defined for the simulation. A CGF system is required to send valid network packets when specified by the protocol; such actions may be time or event triggered. Finally, the arrival of incoming network packets sent by other simulation entities should be handled correctly, as per the protocol. For example, if a CGF system receives a packet that signifies that one of its controlled entities has been hit by an anti-tank missile, it should assess the damage that may result from that impact.

2.3.2.1.2. Battlefield environment simulation

The entities controlled by the CGF system exist in a battlefield which is a simulated subset of the real world. As such, the CGF controlled entities should obey the laws of physics relevant to the activities occurring in the battlefield. Often this means that one must use physical laws to model such behavior Barr[1989], although lower-cost solutions may sometimes be appropriate (i.e. physical modeling vs. simplified kinematics). Using physics, the vehicle dynamics of the CGF entities can be modeled, including acceleration, deceleration, turn rates, and vehicle performance characteristics. (For an example of CGF aircraft vehicle dynamics modeling, see Cimini et. al.[1992].)

The CGF system usually includes a terrain database over which the battle will be fought; it may be a detailed polygonal representation of an actual piece of terrain, or a large featureless plane corresponding to the surface of the ocean. The effects of the terrain on the simulation events should be modeled, including terrain effects on movement and line of sight obscuration.

Because the world being simulated is a battlefield, combat interactions need to be modeled in accordance with the physics of weapon and armor performance characteristics. For example, in both DIS and SIMNET, a CGF tank that fires on a hostile vehicle determines if a hit was achieved using a set of factors that include range, exposure of the target, and performance of the tank's weapon and sighting systems. If a hit is inflicted, the impacted vehicle considers

munition type, range, impact angle, and armor protection to assess the damage it suffers. The accuracy of these calculations is of central importance to the validity of the simulation.

2.3.2.1.3. Support for multiple entities

CGF systems typically provide simultaneous support for multiple entities. Their usefulness is due in large part to this characteristic. The CGF system's architecture must provide a means to allocate processing resources to all of its supported entities.

2.3.2.1.4. Autonomous behavior generation

A CGF system will use built-in behavior to react autonomously to the simulation situation or to carry out orders given by its operator. Its behavior may be encoded as algorithms, production rules, formal behavior specifications, or some other form. The intent is for the CGF system's behavior to be autonomous (i.e. not requiring human control) and realistic (i.e. true to doctrine, physics, and human responses) to the greatest possible extent.

Most current CGF research is focused in the area of autonomous behavior generation. IST and other research labs are attempting to increase the level of autonomy of CGF systems.

2.3.2.1.5. Operator control of behavior

In addition to the autonomous behavior, a CGF system should include an operator interface that allows a human operator to control the CGF entities. The operator may override autonomously generated behavior, or he or she may initiate and control behavior in situations that are beyond the CGF system's capabilities. Existing CGF systems typically provide a map display of the battlefield that shows the battlefield terrain and the simulated entities on it, together with a human command interface.

2.3.2.1.6. Representation of the military organizational hierarchy

Military units have hierarchical organizations. As CGF systems are designed to control larger numbers of entities, it becomes increasingly important to represent the military hierarchy of those entities in the system. With the representation of the hierarchy in place, the operator can give orders to higher level units, or the CGF system can autonomously generate behavior for a unit. Then, the unit level order could be automatically interpreted and passed down to the constituent entities for execution.

2.3.3. The IST CGF Testbed

Under the sponsorship of ARPA and STRICOM, IST has been conducting research in the area of CGF systems, seeking to increase the realism and autonomy of CGF behavior. A key product of that sponsorship is the IST CGF Testbed. The IST CGF Testbed is a CGF system that provides an environment for testing CGF behavioral control algorithms. It is documented in Danisas et. al.[1990], Gonzalez et. al.[1990], Petty[1992c], Smith et. al.[1992a], and Smith et. al.[1992b].

A minimal IST CGF Testbed configuration consists of two standard IBM-compatible personal computers (PCs), each running one of the two software components of the CGF Testbed; the "Simulator" and the "Operator Interface" (or OI). The Simulator performs the computations for vehicle dynamics, battlefield environment simulation, and behavior generation for the computer controlled CGF entities. The OI provides an operator interface to the CGF system consisting of a plan view display battlefield map and a menu-based mouse and keyboard

input system. The CGF operator enters commands for the CGF entities using the OI, which passes those commands to the Simulator for execution by the CGF entities. The OI and the Simulator communicate via the simulation network.

One of the goals of this research on CGF systems was to demonstrate the feasibility of low-cost CGF systems. Consequently, the IST CGF Testbed was developed and runs on IBM-compatible personal computers. The basic configuration can support approximately 12 CGF entities, which is roughly the number of vehicles in a military unit of company size.

The simulation network is used for communication between the Simulator and the OI was to permit easy scaling of the Testbed. If more than 12 entities or more than one OI is required, additional PCs may be attached to the network. The Simulator and OI software both adjust automatically to the presence of more than one of either component.

The Testbed software is written in ANSI C, and is compiled with the Borland C++ compiler. The IST CGF Testbed can be compiled so as to communicate on the simulation network using either of two network protocol standards: DIS 2.0.3 or SIMNET 6.6.1.

Like other CGF systems, the IST CGF Testbed depends on the operator to perform high level planning and mission control. However, the IST CGF Testbed contains a range of flexible intermediate level behaviors that operate automatically and realistically.

One example of such intermediate level behavior is the Testbed's Fire_Weapons behavior for infantry fighting vehicles (IFVs). IFVs, such as the M2 Bradley or the Russian built BMP, typically have a variety of weapons systems, each suited to a particular type of target and tactical situation. M2 Bradleys are armed with a coaxial machine gun, used against infantry at short range; a 25mm cannon, used against infantry at longer range and vehicles other than tanks at short and medium range; and TOW anti-tank missiles, used against vehicles other than tanks at long range and tanks at all ranges. The IFV Fire_Weapons behavior performs the following actions without operator intervention:

1. Scan the terrain for visible enemy targets.
2. Select the most threatening enemy target from among those visible, based on threat analysis rules encoded in the behavior.
3. If more than one enemy target falls into the same threat category, select one from among those available based on a fire distribution scheme that considers nearby friendly entities.
4. Select the appropriate weapon for that target.
5. Prepare the weapon for firing (aim the turret and possibly raise the TOW launcher).
6. Fire the weapon and determine if a hit was scored.
7. Reload the weapon.

For low level behavior, the CGF Testbed includes several fast and effective algorithms. For example, the route planner will find routes around terrain obstacles.

The primary means of behavior specification of the IST CGF Testbed is a code structuring technique based on finite state machines (FSMs). Behavior in the Testbed is encoded as algorithms, written in C. However, that C code is given structure using the FSM mechanism. The basic idea is that atomic units of behavior, implemented as C functions, become states in a FSM. Each state determines the next state to be entered by testing simulation conditions; thus transitions may be triggered by simulation events. More complex behavior can be constructed, bottom up, by combining simpler FSMs. FSMs exist as actual structures in the CGF Testbed, with each state containing a pointer to the function corresponding to the state. The FSM mechanism is described in detail in Smith et. al.[1992a] and Smith et. al.[1992b].

3. Intervisibility concepts

What does intervisibility mean in a simulation environment? What effect does the rate at which we perform intervisibility updates have on the simulated entities and the simulation in general? These are some of the questions that this report attempts to answer. This section describes some of the basic concepts for intervisibility.

3.1. Definitions

Before the question "Does an entity see another entity?" is answered within a CGF system, the question "Is it possible for the entity to see the other entity?" must be answered? Typically, CGF systems' "sighting models" determine if entities "see" one another and incorporate the effects of attention, angle of view, weather, obscurants, time of day, and the details of sensor systems. However, sighting models are active only after "intervisibility" between two entities has been established. An intervisibility determination establishes or denies the existence of an unblocked Line of Sight (LOS) between two entities.

Establishing intervisibility between two entities involves checking to see whether terrain features (such as hills) or other objects (such as other vehicles) prevent a ray of light from traveling from one entity to the other. If the ray of light travels unhindered (by terrain features or other objects) the two entities are said to have "unblocked" intervisibility else the intervisibility is said to be "blocked." For the remainder of this discussion, the term "see" refers only to an unblocked LOS and does not imply that a sighting model has determined that an entity has sighted another.

It is worth noting that intervisibility, as used in this discussion, is symmetric; if A can see B then B can also see A (the direction an entity is facing is not considered). Although intervisibility is also reflexive (A can see A) intervisibility is ***not*** an equivalence relation because it is not transitive (A can see B and B can see C does not mean A can see C).

The phrase "intervisibility update" is used when an entity is invoked to determine which of its opponents it can see. If A performs an intervisibility update and there are n opponents in the scenario, A will do n individual (i.e., A-to-B) intervisibility determinations. The intervisibility update for an entity is initiated by the entity periodically receiving an "intervisibility update message." The periodicity of these messages is determined by the "intervisibility update rate" (IUR).

Whenever the intervisibility status changes (from blocked to unblocked or vice-versa) an intervisibility "transition" is said to occur. For example, a vehicle may appear from behind a hill becoming visible to another, thus causing the intervisibility status between them to transition from blocked to unblocked.

3.2. Computational methods and cost

Three algorithms to efficiently conduct an intervisibility determination between two entities were previously developed and evaluated at IST (Petty et. al.[1992a]):

1. Algorithm F: Grid/edge method
2. Algorithm C: DCEL Traversal method and
3. Algorithm P: Triangle Traversal method.

Algorithm F is in use in the IST CGF research Testbed; a brief description of the intervisibility determination algorithm and its time complexity is given in Section 4.1.

Since extensive terrain checks are required to determine intervisibility status (blocked or unblocked) and the checks themselves are complicated, it comes as no surprise that the intervisibility computation taxes a system's resources. Very high intervisibility rates (more intervisibility updates per second) have shown to load a CGF system so much that the generated vehicle behavior degrades. For example, vehicle behavior may degrade such that routes are not followed as planned, possibly leading to vehicles circling. Thus, it is important that the time spent in performing intervisibility determinations be reduced without sacrificing the tactical behavior of vehicles. New algorithms may be invented to make the intervisibility determination process take less time or new heuristics may be employed to reduce the number of intervisibility determinations; the latter procedure should have minimal, or no, effect on the behavior of the system.

The time consumed by the intervisibility procedure is not the sole cost of doing intervisibility determinations. Internal system messages to initiate intervisibility updates require handling and delivery. If intervisibility update rates are increased, the delivery costs may become significant.

3.3. Statement of the problem

As already discussed, intervisibility determinations can use a lot of system time which makes attempts to reduce intervisibility determinations worthwhile. Inventing new algorithms for intervisibility determination can be very difficult. Highly optimized intervisibility determination algorithms rely on the format of the underlying terrain database and may become useless if the underlying structure of the terrain database needs to be changed. For example, switching from a polygonal to a spline-based terrain representation would render polygonal-based intervisibility algorithms useless.

The goal of the research described in this report is to reduce the total computational load of intervisibility determinations on a CGF system. The reduction is to be achieved in a manner that has a minimum impact on the realism of the CGF entities' behaviors generated by the system.

IST's previous intervisibility research focused on efficient intervisibility determination within a polygonal terrain database. For the current work, an attempt was made to reduce the number of intervisibility determinations made by a CGF system. The approach taken was to design, implement, and evaluate heuristics which would decide which intervisibility determinations could be skipped or delayed without affecting the CGF entities' behavior. An important characteristic of this heuristic approach is that it is independent of terrain database format. As a result, the heuristics described and evaluated herein can be used in any CGF system.

4. Intervisibility in the IST CGF Testbed

This section discusses the way intervisibility is done in the IST CGF Testbed and the data structures used to track entity intervisibility status, i.e., who has intervisibility to whom.

Section 4.1 gives a brief description of the algorithm in use in the IST CGF Testbed.

Section 4.2 discusses the sightings list and the intervisibility update duration (IUD).

Section 4.3 discusses the modifications made to the system to implement the heuristics and collect data to evaluate them.

4.1. Intervisibility determination algorithm

Algorithm F (Petty et. al.[1992a]), the Grid edge traversal method algorithm, is used in the IST CGF Testbed for computing intervisibility between entities. The CGF Testbed's internal terrain database format, which is based on the SIMNET format, is a polygonal database. The surface of the terrain is represented in the database by contiguous non-overlapped polygons; the 3D vertices of each polygon are used to compute the height of any point within the polygon. The terrain database is divided into 500 meter x 500 meter squares called *patches*. Each patch is composed of sixteen 125 meter x 125 meter squares called *grids*.

This sub-division of the terrain database into patches and grids ensures that an algorithm can access and traverse the terrain with ease. Both patches and grids can be referenced in the data structure by computing indices based on the coordinates of a point inside them. Polygons are organized by their patch and grid locations. Each edge within a patch structure has a code which tells which grids within the patch contains that edge. Polygons do not span patch boundaries. Further, polygons are defined such that patch boundaries coincide with polygon edges, although the reverse is not true (polygon edges do not have to coincide with a patch boundary).

The first step in the determination of intervisibility is *point location*, i.e., locating the patch and grid in the terrain database containing the sighter and target locations; the time for this operation is a constant independent of the location of the points.

After point location is done, the patches and grids through which the Line Of Sight (LOS) from the sighter to the target passes are determined; these patches and grids must be searched for possible intervisibility blockages by base, treeline, or canopy polygons. Patches and grids are determined by a traversal across the terrain database using Bresenham's algorithm (Bresenham[1965]). As each patch and grid on the Line of Sight is identified, it is systematically searched to determine if any terrain feature (base terrain, treelines, buildings, etc.) on it blocks the intervisibility.

This process is computationally expensive because all the edges of the polygons in the grids containing the LOS must be checked to see whether they intersect the LOS in two dimensions. If a polygon edge/LOS intersection is found, the algorithm calculates whether the intersection lies above the LOS or below it in three dimensions. If the intersection is above the LOS, the LOS is blocked by the polygon edge; otherwise the LOS passes above the edge and is unblocked by that edge.

The large number of polygon edge/LOS tests, the complexity of determining the intersections, and establishing whether the intersections lie above or below the LOS produce a time

consuming computation. The time complexity of the algorithm is expressed as the sum of the times to determine:

Point location:
O(1) time (simple arithmetic on the point's coordinates suffices).

Line intersection:
O(e) time per grid, where e is the number of polygon edges in the grid.

Features:
O(m) time per grid, where m is the number of feature segments in the grid.

4.2. The sightings list and intervisibility update duration

Each entity in the simulation keeps track of its sighting status to other entities by maintaining a *sightings list*. A sightings list is a linear array that is a snapshot of the intervisibility status in the simulation for an entity.

The sightings list is a doubly linked list containing SIGHTINGS_ENTRY records. Each SIGHTINGS_ENTRY record describes the entity on whom sighting is being done, i.e., the target entity. It contains a pointer to the target's "dead reckoning model" which describes its "dynamics" information, for example, its velocity. Other fields in the SIGHTINGS_ENTRY contain the time when the target entity was last sighted, the current visibility status, and the highest visibility status attained so far. The visibility status indicates intervisibility status, i.e., LOS_ERROR, LOS_INVISIBLE, LOS_DETECTED, LOS_RECOGNIZED and LOS_IDENTIFIED. These levels refer to how well the entity can see the target entity.

During an intervisibility update, an entity does point-to-point intervisibility determinations to all target entities within visual range. Depending on the result of the point-to-point intervisibility determination (can the target entity be seen?) the status of the target entity is updated. New entities (those not already on the sightings list) are added to the list. Target entities which have remained invisible for a time greater than the "sighting persistence" limit are removed from the sightings list.

Receipt of an intervisibility update message triggers an entity to do its intervisibility update. When the intervisibility update is completed, the entity sends itself another intervisibility update message. Table 4.2-A gives a description of the terms used in connection with intervisibility update durations and rates.

Table 4.2-A Intervisibility update durations and rates.

Term	Description
IUD	Intervisibility Update Duration. This is the time between successive intervisibility updates and can vary as the simulation proceeds. In the IST CGF Testbed this is measured in one hundredths of a second. For example, an IUD of 25 means that intervisibility updates would be done every 0.25 seconds. A lower bound on the IUD depends upon the simulation load and the type of hardware that is used for the simulation.
IUR	Intervisibility Update Rate. This is the frequency of the intervisibility updates and is the inverse of IUD. Thus, an IUD of 25 yields an IUR of 4 (i.e., 4 updates per second).
BUD	Base Update Duration. The IUD is initially set to this value which is read from a configuration file (sim.lod). The BUD is a constant for the simulation and is also measured in one hundredths of a second. A lower bound on the BUD depends upon the simulation load and the type of hardware that is used for the simulation.
BUR	Base Update Rate. This is the initial frequency of the intervisibility updates and is the inverse of the BUD. It remains constant throughout the simulation.

Ideally, an IUD of 25 would cause an entity to do 4 intervisibility updates per second (an IUR of 4), while an IUD of 400 would, ideally, cause the entity to do 1 update every 4 seconds (an IUR of 0.25). If we are using an IUD of 100, we expect updates to be done at t+1 seconds, t+2 seconds, etc. The "ideal" IUR is always a maximum because of the systems clock granularity and various kinds of system overhead.

Assuming an intervisibility update requires Δt seconds and the IUD is t seconds, the intervisibility updates would occur every $t+\Delta t$ seconds without adjustment. To maintain the expected rate, messages should be queued with delays of $t-\Delta t$ seconds rather than t seconds. Even this will fail to establish the desired rate because messages may be arbitrarily delayed in an overloaded system. It should be noted that the standard IST CGF Testbed makes no such adjustment; the user specifies a duration and this is used as specified. The actual intervisibility update rate varies with the intervisibility algorithm and system load, but is always less than 100/IUD.

Although at lower IURs (i.e., high IUDs) these delays are insignificant, one type of heuristic to be examined later Continuous Intervisibility Determination Avoidance (CIDA), Section 5.1.5, requires precise message delays because of the high message rates involved. Messages in the system are always "late" because of other messages ahead of them in the message queue, the computer's clock granularity, and other factors. The high message rate requested for CIDA heuristics exacerbates this problem even further. In order to achieve the message rate for these heuristics, the software makes an adjustment to the message delay.

4.3. Modifying the IST CGF Testbed's intervisibility process

To accomplish the research objectives of this effort, several changes to the IST CGF Testbed had to be implemented. For example, the intervisibility heuristics implemented require some historical information to avoid going arbitrarily long without doing an intervisibility determination. To retain historical and other heuristic specific information, an intervisibility "scratch" area was added to the dead reckoning structure for use within the intervisibility code. The structure of this data is known only within the intervisibility code and varies with the heuristic. All of the heuristics keep an indication of the last time an intervisibility determination was done.

The standard CGF Testbed Simulator contains some intervisibility heuristics and a sighting model used to filter sightings based on human vision. These elements of the Simulator were removed as only a physical intervisibility sighting was of interest (the sighting model depends on the intervisibility code, not the other way around).

Other minor modifications to the CGF Testbed were made to allow statistics about the intervisibility determinations to be automatically gathered.

5. Intervisibility heuristics

The intervisibility heuristics implemented as part of this project and their characteristics are represented by Table 5-A. A "X" in a column means the characteristic is supported by the implemented heuristic. Note that discrete heuristics are also known as coarse-grain heuristics and continuous heuristics are also known as fine-grain heuristics.

Table 5-A Implemented heuristics and their characteristics.

Heuristic	Characteristic		
	Composite	Discrete (coarse-grain)	Continuous (fine-grain)
Varying the BUR			
Symmetry (Sym)		X	
History (His)		X	
Sighter-based (Sgt)	X	X	
Target-based (Trg)	X	X	
Fine-grain history (Fgh)			X
Fine-grain sighter (Fgs)	X		X
Fine-grain target (Fgt)	X		X

Section 5.1 discusses the different types of heuristics.

Section 5.2 discusses the intervisibility update rate (IUR) limits associated with discrete and continuous intervisibility determination avoidance heuristics.

Section 5.3 discusses the overheads incurred in employing a heuristic and processing additional messages (for discrete and continuous heuristics).

Section 5.4 gives a description of the implemented heuristics and the rational behind them. Some key variables used by these heuristics and how they vary as the simulation proceeds is discussed there.

5.1. Types of heuristics

Generally, intervisibility heuristics are of two types: the *physical* and the *behavioral* heuristics. Physical heuristics attempt to reduce the number of intervisibility determinations by using some physical characteristic of intervisibility whereas behavioral heuristics exploit vehicle behavior in their attempt to reduce the number of intervisibility determinations.

A good example of a physical heuristic is the symmetry heuristic (discussed in Section 5.1.2). It is based on the fact that due to the nature of light if A can see B then B can see A. Thus, the physical nature of light is the basis behind this heuristic. In a literal sense this is not really a heuristic because a heuristic by definition involves some "guessing" or approximation. There is no guessing about symmetry because the intervisibility relation is symmetric.

Behavioral heuristics attempt to reduce the intervisibility determinations by using some behavior being done by an entity; for example, a destroyed entity need not update intervisibility because having intervisibility to a destroyed vehicle is of no interest to the Simulator. The various coarse-grain and fine-grain heuristics are all behavioral heuristics.

5.1.1. Varying the base update rate

As already mentioned (Section 4.2), an intervisibility update for an entity is initiated by the receipt of an intervisibility update message. Updates may be made more frequent by increasing the frequency of these messages. The frequency is controlled by the intervisibility update duration (IUD). The IUD is initially set to the base update duration (BUD), which is defined by the user in a support file (sim.lod)

In situations where a battle is fought in obstructed or hilly terrain, a slow intervisibility update rate may have a deleterious effect on the scenario. For example, sightings that could have been possible during periods of brief intervisibility may be missed because of a low IUR. Hence, entities that could have been destroyed earlier may survive to inflict damage on their potential destroyers, thus changing the scenario altogether.

5.1.2. Symmetric heuristic

As already noted, intervisibility is symmetric (if A can see B then B can see A) and, equally important, lack of intervisibility is also symmetric (if A cannot see B then B cannot see A). An obvious heuristic is to inform B when A determines (through a point-to-point intervisibility determination) that $A \mapsto B$. The symbol "$\mapsto$" stands for "can see". This result is kept in the intervisibility scratch area for B. When B does an intervisibility determination, it first consults the scratch area for a previously generated result. Stored along with the result is the time at which $A \mapsto B$ was determined. The time must be recorded because there may be a time interval between when A determines the A-to-B result and when B references that result to determine whether $B \mapsto A$. If the interval is "large", it is assumed that A or B has moved enough to invalidate the determination and it is recalculated. If the interval is "small", the result supplied by A is accepted. It is this time element that makes symmetry a heuristic. Without it $A \mapsto B$ would always means $B \mapsto A$.

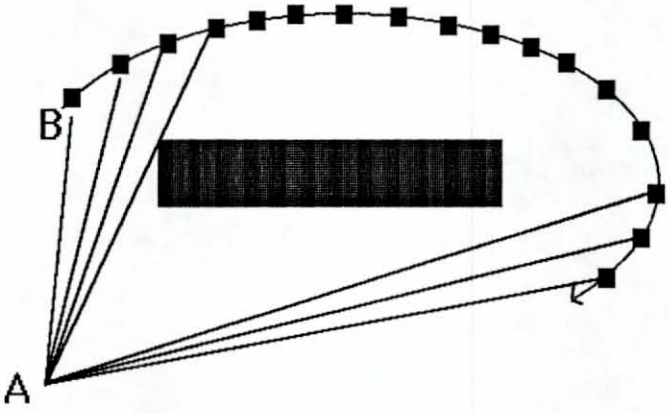
5.1.3. History heuristic

It is often the case that when the outcome of an experiment or observation is consistent over time, people begin to assume the next outcome or observation will be the same. If a die is cast many times and always turns up 6, observers will eventually become convinced the result will continue to be 6 and will stop observing the rolls. This is not necessarily a bad thing; in the case of the die it is likely the die has a gimmick. In survival situations, spending less time on one activity frees time for other actions.

For the problem at hand, it is reasonable to assume that if we have had intervisibility to a target for a "long" time, we will still have intervisibility the next time we check. In such a situation, we are likely to be operating in the immediate vicinity of the vehicle of interest. Similarly, if we have not had intervisibility to a vehicle for a "long" time, it is likely we will not have intervisibility the next time we check. In this case, we may be impossibly distant from the target.

The history heuristic is based on these ideas. The heuristic tracks the number of consecutive intervisibility determinations that have returned the same intervisibility value. When a threshold value is passed, intervisibility determinations are skipped effectively reducing the update rate. When a transition is made (sighted to not-sighted or vice versa), skips are again inhibited until a sufficient history accumulates.

Consider the following scenario:



In this case, A observes B four times before it is obscured. B is then not in sight for 10 updates, then comes back into sight for 3 (transitions come in bursts). In this case 2 transitions occurred in 17 checks. In actual combat situations much more favorable ratios (fewer transitions per check) can be expected, since entities may be kilometers apart with many intervisibility obstacles.

It can be argued that this technique simply decreases the IUR. If transitions are rare this is true: a transition will occur and skipping will engage before the next transition. Hence, the intervisibility check rate is effectively lower at the time of transitions. On the other hand, if transitions are common, the IUR will increase at the first transition and repeated transitions will keep the rate high.

In actual combat situations, it is assumed transitions (as shown in the diagram above) will come in bursts (several transitions followed by long gaps without transitions). In such a situation, there may be some delay for the first transition detection, but the others will be rapidly recognized.

Whereas the discrete history heuristic simply produces a binary result (skip, do not skip), the fine-grain mechanism allows the heuristic to produce a range of values which are mapped to the number of skips to be done.

5.1.4. Discrete intervisibility determination avoidance heuristics (DIDA)

The basic idea is to reduce intervisibility computation by skipping some of the intervisibility determinations for an entity. The role of the heuristic is to decide when to skip.

For discrete avoidance the IUR, which had been initially set to the BUR, is not modified. Intervisibility rate reduction is accomplished by skipping selected A-to-B intervisibility determinations. The main fault with this technique is its granularity. If the BUR is set as low (i.e., as infrequent) as is practical for realistic behavior, any skip (unless cunningly selected) is apt to cause behavior deterioration due to missed sightings. A natural tendency is to increase the BUR when discrete avoidance techniques are employed. DIDA heuristics are also known as coarse-grain heuristics.

Boolean heuristics are desired for discrete intervisibility determinations. The only choice is either to skip or not to skip the A-to-B intervisibility determination under consideration. Many heuristics lend themselves to this approach. For example, they are along the lines of

"skip if we did not skip the last check, we got the same result the last M times we checked, and we did not see an enemy approaching in the last N checks." Of course, the value returned when a skip is done should agree with the last value returned. It is unlikely we can predict a transition from not-sighted to sighted or vice versa.

5.1.5. Continuous intervisibility determination avoidance heuristics (CIDA)

Continuous avoidance means that the interval between intervisibility determinations (IUD) is not necessarily an integer multiple of the BUD. The heuristics guess how long it is safe to wait until the next intervisibility determination based on the behavior of the entities in the simulation. CIDA heuristics are also known as the fine-grain heuristics.

An optimal continuous avoidance algorithm would require separate intervisibility update messages for each A-to-B intervisibility determination being considered. Because the time for the individual intervals could be any positive value, optimal delays could be used for the various intervisibility determinations. This approach was not seriously considered for this project because of technical problems in starting up and shutting down such a process, and the considerable programming and run time overhead involved.

Another approach is to adjust the interval between intervisibility update messages. This has the disadvantage of delaying a complete intervisibility update by an entity in order to delay a specific A-to-B intervisibility determination. In cases where all of the A-to-B intervisibility determinations are willing to wait, the update rate may be slowed (at least temporarily). This approach was not used in these experiments.

One practical technique is to approximate continuous delays for individual intervisibility determinations by using a relatively high IUR (in comparison to the BUR) in combination with the discrete techniques. This is called "fine-grain" check avoidance.

This approach is supported in these experiments. With suitable parameter adjustments (no algorithmic changes), the fine-grain technique will delay most intervisibility determinations most of the time and "intelligently" select which A-to-B intervisibility determinations should be applied for a given intervisibility message event (or an intervisibility update). With a high IUR, a reasonable approximation of arbitrarily selected delays between A-to-B intervisibility determinations is feasible.

For continuous avoidance, the heuristics used should yield a continuum of values. For convenience, all such functions are constrained to yield a range from zero to one. Zero indicates a minimum delay should be used (another intervisibility determination is needed *soon*), whereas one indicates the next intervisibility determination may be delayed by the maximum allowed interval. The latter remark indicates a primary assumption of the project, *every heuristic has a maximum allowed delay* (which is the same as a minimum update rate; see Section 5.2).

The IUR should be high enough to approximate continuous delays substantially better than the BUR. But care should be taken that a high IUR (a low IUD) does not overload the system to the extent of degrading the behavior. Moreover, these changes should be transparent to the user who expects the checks to happen at the BUR.

In these experiments the IUR is obtained by multiplying the BUR by a *RATE* factor. This value indicates the number of times intervisibility checks are actually requested by the system as a multiplier of the BUR. Appendix A.4 discusses why a RATE value of 4 was used for the CIDA or fine-grain heuristics.

5.1.6. Composite heuristics

The composite heuristics are composed of sub-heuristics which vote whether to do an intervisibility determination. Each sub-heuristic computes a *metric* (*M*) value based on certain characteristics of the current simulation state. The computed metrics for the various sub-heuristics are used to determine a weighted average. If the weighted average exceeds a *threshold,* an intervisibility determination will be skipped.

Sighter and target-based heuristics (Table 5-A) lend themselves to being composite heuristics. Composite heuristics can be either discrete (coarse-grain) or continuous (fine-grain). For example, the sighter-based heuristics occur as discrete ("Sgt") or continuous ("Fgs") versions. Whether a heuristic is composite or not is independent of whether it is discrete or continuous.

5.2. Update rate limits for heuristics

This section discusses the intervisibility update rate limits for CIDA and DIDA heuristics.

Although heuristics could make recommendations for arbitrarily long delays between updates, it is unlikely that very low IURs would be effective. None of the heuristics designed could be expected to correctly predict long transition free periods (unless a vehicle is destroyed).

To prevent serious sighting delays, all the heuristics use a minimum sighting rate of *half the user's requested rate* (BUR), unless a vehicle is destroyed; some heuristics prevent further sighting to or by destroyed vehicles. Given a BUR of 4.0, this ensures that after a sighting at t_0 another sighting will occur no later than (t_0+0.5) seconds.

5.2.1. Discrete heuristics

The coarse-grain heuristics receive update messages at the same rate as the user specified (the BUR). Using the scratch area, these heuristics ensure that only one intervisibility determination is skipped for any two consecutive messages. Thus, for a BUR of 1.0, if an intervisibility determination is done at t_0, the next determination will occur at either (t_0+1.0) or (t_0+2.0).

5.2.2. Continuous heuristics

This section discusses the minimum, maximum, and possible intervisibility update rates for continuous heuristics.

5.2.2.1. The minimum update rate for continuous heuristics

The intervisibility scratch area is used to keep an indication of the last time an intervisibility determination was accomplished. When an intervisibility message arrives this field is checked and, if necessary to avoid too great a gap between updates, an intervisibility determination is forced. When this happens in a fine-grain heuristic, the heuristic is invoked to determine a new interval before the next determination. For example, when a fine-grain heuristic recommends a delay that would give too low an IUR, an intervisibility determination will be forced at the minimum sighting rate (see Section 5.2). On the other hand, if the heuristic demands a determination sooner than that required by the minimum sighting rate, the request is honored up to the point described in Section 5.2.2.2.

5.2.2.2. The maximum update rate for continuous heuristics

Although it is reasonable to hope heuristics could predict times when an accelerated rate would be beneficial, that is not a goal of this project. To this end, heuristic's recommendations are *always bounded above by the user requested rate* (BUR). This ensures the CGF Testbed Simulator will not attempt to do more frequent intervisibility determinations than the base system. For example, given a BUR of 2.0, after a sighting at t_0 another sighting will occur no sooner than (t_0+0.5) seconds.

5.2.2.3. Possible update intervals for continuous heuristics

For all the experiments completed, the fine-grain heuristics received messages at four times the rate required by the BUR. So, for a BUR of 1.0 messages per second messages were received (approximately) every 0.25 seconds. Given this, and the restrictions on the update rates, after an intervisibility determination at t_0, the next determination will occur at either (t_0+1.0), (t_0+1.25), (t_0+1.5), (t_0+1.75), or (t_0+2.0) seconds.

To summarize, all the fine-grain heuristics contain code to ensure that intervisibility determinations are not done more often than the user. Removing this limiting code would cause the Simulator to do additional determinations when recommended by the heuristic, possibly increasing realism. *However, the objectives of these experiments is to reduce the number of intervisibility determinations. Hence, these experiments bound the number of determinations at or below the user requested rate.*

5.3. Heuristics and message overhead

The fine-grain technique increases the message handling overhead by increasing the number of intervisibility update messages sent. The finer the granularity, the greater the overhead.

On an individual Simulator, the number of additional messages, for a fixed granularity, is bounded. New remote vehicles, vehicles that have been created on other Simulators, add no new messages. Only the addition of local vehicles, vehicles that have been created on this Simulator, adds to the message load.

The number of messages per unit time is easily computed as the message rate per vehicle times the number of vehicles. However, for the fine-grain updates the message rate is *not* the value requested by the user in the Simulator's configuration files (the BUR).

The fine-grain heuristics are designed to increase the message rate according to a multiplier. When this parameter is 1 no additional messages are generated. When it is R, the requested rate will approximate R times the user requested rate (because of delivery overhead, which increases with R, the actual rate will be less than expected, and the discrepancy will increase as R is increased). For these experiments, R was set to 4 (see Section 5.2.2.3).

5.4. Implemented heuristics

The heuristics implemented are:

- Varying the intervisibility base update rate (BUR),
- The symmetry heuristic,
- The history heuristic (in coarse and fine-grain versions), and
- The composite heuristics (in coarse and fine-grain versions)

5.4.1. Varying the intervisibility base update rate

This is implemented as discussed in Section 5.1.1

5.4.2. Symmetry heuristic

The symmetry heuristic is implemented as discussed in Section 5.1.2.

5.4.3. History heuristic

For a discussion of the history heuristic see Section 5.1.3. This section discusses the mechanism that recommends whether to skip or not.

Both the coarse-grain and the fine-grain version of the history heuristic track the number of consecutive intervisibility determinations which have returned the same intervisibility value. In the coarse-grain version, the history of identical intervisibility values are compared to a threshold value. When the threshold value is exceeded, intervisibility determinations are skipped effectively reducing the update rate. When a sighting transition is made (sighted to not-sighted or vice versa), skips are inhibited until a sufficient history accumulates. The layout of the code ensures that the simulation doesn't go too long without doing an intervisibility determination between the two entities.

The fine-grain version calculates a skip value after a sufficient sighting history accumulates; this value refers to the number of intervisibility determinations that can be safely skipped between a sighter and a target. Again, the layout of the code ensures that the simulation doesn't go too long without doing an intervisibility determination between the two entities. As is usual for the fine-grain heuristics, an upper bound on the IUR is maintained to prevent checking more often than the user requested rate.

The formula used to calculate the skip value in the fine-grain version is:

$$Skip = RATE - 1 + (MAX\_SKIP - RATE + 1) \bullet \frac{Matches}{Interval}$$

where:

Skip	is the number of fine-grain intervisibility determinations to be skipped for this sighter/target pair
RATE	is the fine-grain multiplier (IUR = *RATE* · BUR)
MAX_SKIP	is the maximum number of skips to preserve the minimum update rate (*MAX_SKIP* = 2 · *RATE* - 1)
Matches	is the number of consecutive intervisibility determinations yielding the same result
Interval	is the "full confidence interval". Once *Matches* equals *Interval* the minimum update rate is used. This is a heuristic parameter.

When there have been no matches, Skip will be *RATE* - 1, which yields an intervisibility update rate equal to the BUR. If the number of matches is as great as the interval, a heuristic parameter, the maximum number of skips will be done. The layout of the heuristic code prevents Skip values exceeding *MAX_SKIP* from delaying intervisibility determinations beyond the *MAX_SKIP* value.

5.4.4. Composite heuristics

The composite heuristics and the sighter and the target-based heuristics are discussed in the following sections. Recall that each component of a composite heuristic always computes a metric value (*M*) based on certain characteristics of the current simulation state. The metric computed by each sub-heuristic lies in the interval [0,1]. Values exceeding 1 are mapped to 1 while values below 0 are mapped to 0. Section 5.4.4.1 discusses sighter-based sub-heuristics and give an equation to compute the metric (*M*). Section 5.4.4.2 discusses the target-based sub-heuristics and give an equation to compute the metric (*M*).

Remember that the composite heuristics occur in both the coarse-grain and the fine-grain versions.

5.4.4.1. Sighter-based heuristics

Sighter-based heuristics attempt to reduce the number of intervisibility determinations done by an entity by taking its behavior into account. The behavior may be some physical action of the entity, such as being stationary, or it may be some abstract behavior, such as having permission to fire.

Four sighter entity behaviors were characterized for this study:

1. The movement of the sighter,
2. The sighter's permission to fire,
3. The sighter's ability to fire, and
4. The proximity of the sighter to enemy entities

In both the coarse-grain and fine-grain versions of the sighter-based heuristic, the rate at which the entity does intervisibility determinations is dynamically adjusted.

The four sub-heuristics are assigned weights to increase or decrease their effects in deciding whether an intervisibility determination is required. A weighted average is computed to decide whether to do an intervisibility determination. In all cases, 0.0 indicates to do an intervisibility determination and 1.0 indicates NOT to do an intervisibility determination.

Any weight can be assigned to a sub-heuristic. The heuristic computes a metric for each behavior which is then multiplied by the weight assigned to it. For some behaviors this metric may be a boolean metric (0 or 1). For example, does an entity have permission to fire? For other behaviors this metric may be a floating point value; for example, the metric associated with the distance of a sighter to an enemy entity may be large (small) if the sighter is near (far) from the enemy entity.

The weighted average of the metrics for all the behaviors is guaranteed to be in the interval [0,1]. The weighted average of the metrics is given by the following equation.

$$Weighted\_average = \frac{\sum_{i=1}^{4} w_i \bullet M_i}{\sum_{i=1}^{4} w_i}$$

where:

wi is the weight assigned to a sub-heuristic i and
Mi is the metric computed by heuristic I (Mi is in the interval [0,1])

For the coarse-grain heuristics, judging whether or not to skip an intervisibility determination requires that a split point (threshold) be determined within this range. If the weighted average is greater than the split point, an intervisibility determination should be skipped, whereas, a value less than the split value requires an intervisibility determination.

The fine-grain sighter heuristics are precise analogs to their coarse-grain versions. The key difference is that the composite vote of the heuristic elements no longer needs to be binary (no split point is needed) because the intervisibility determination is delayed by skipping intermediate fine-grain intervisibility determinations.

In this case the number of fine-grain intervisibility determinations skipped is given by:

$$Skip = RATE - 1 + (MAX\_SKIP - RATE + 1) \bullet Weighted\_average$$

where:

Skip	is the number of fine-grain intervisibility determinations to be skipped
RATE	is the fine-grain multiplier (IUR = *RATE* · BUR)
MAX_SKIP	is the maximum number of skips to preserve the minimum update rate (*MAX_SKIP* = 2 · *RATE* - 1)
Weighted_average	is the weighted average of the metrics associated with each sub-heuristic

A weighted average of 0.0 will yield the maximum check rate (*RATE*-1) while a value of 1.0 will yield the minimum rate (*MAX_SKIP*).

5.4.4.1.1. Moving and stationary

This sub-heuristic is based on the premise that moving entities need to check intervisibility more often than stationary entities. A moving person will look around the surroundings more often than a stationary person.

This sub-heuristic requests the minimum rate for a stationary vehicle and the maximum rate for a vehicle moving at its "normal speed".

$$M_1 = \frac{normal - current}{normal}$$

where:

normal is the normal speed of a vehicle (for example, 10 m/s.)
current is the current speed of the vehicle (in m/s.)

When a vehicle is stationary, its "current speed" is 0 and the value of the metric, M_1, is equal to 1. This means that the sub-heuristic requires an intervisibility determination to be

skipped. As the vehicle builds up speed, its "current speed" rises with a corresponding decrease in the value of M_1. When the "current speed" equals the "normal speed", M_1 is 0 signifying that the sub-heuristic does not require any intervisibility determination to be skipped.

5.4.4.1.2. Permission to fire

It seems reasonable that entities that have permission to fire should conduct more intervisibility determinations than entities that do not (provided they are not destroyed). This sub-heuristic falls into the category of boolean sub-heuristics because an entity either does or does not have permission to fire.

M_2 = 1.0 if entity has permission to fire, 0.0 otherwise

5.4.4.1.3. Able to fire

The ability to fire may be lost by an entity after it has been hit by enemy fire. This sub-heuristic also falls in the category of boolean behaviors.

M_3 = 1.0 if able to fire, 0.0 otherwise

5.4.4.1.4. Proximity to target

It seems natural that an entity would do more intervisibility determinations when it is in the vicinity of enemy entities than when it is far away from them. The entity is considered "in the range" of an enemy entity if is lies within the maximum range of any weapon possessed by the target entity.

As the entity or sighter gets closer to the enemy or target entity more intervisibility determinations are done.

$$M_4 = \frac{d}{r}$$

where:

d is the distance to the target entity
r is the maximum range of any weapon possessed by the sighter entity

5.4.4.2. Target-based heuristics

Target-based heuristics reduce the number of intervisibility determinations done by a sighter to *a particular* target by taking into account the type, appearance and behavior of that target. Moreover, there is provision in the coarse-grain version to ignore targets that have been destroyed or have lost firepower and, thus, eliminate even the overhead of computing the weighted average.

The implementation of the target based heuristics is very similar to the sighter-based heuristics except characteristics of the target are examined.

Four target entity behaviors were characterized for this study:

1. The relative movement of the sighter and target
2. The estimated threat of the target
3. Target damage status and

4. The proximity of the sighter to a target

5.4.4.2.1. Relative movement of sighter and target

The movement dimension to this heuristic uses both the sighter's and target's velocities to determine whether the vehicles are closing or separating and at what speed. The rational is that when the entities are closing, the sighter needs to more carefully watch the target.

$$M_5 = 0.5 + \frac{R}{2 \bullet C}$$

where:

R is the rate at which the sighter and target are closing (in m/sec.), expected range $[-C, C]$

C is the closing rate at which maximum update rate is to be used (12 m/sec. in these experiments)

If vehicles are separating rapidly $R \leq -C$, yielding $M_5 = 0$. If they are closing rapidly, $R \leq C$ and $M_5 = 1$.

5.4.4.2.2. Estimated threat of target

Sighters do more intervisibility determinations to targets that are considered more threatening than to targets that are less threatening.

The target's threat is determined by examining the priority the target has been given as a target for the weapons the sighter carries; these priorities are configured by the user. If the target is top priority for any of the sighter's weapons, frequent intervisibility determinations are recommended (value of 0.0). If the target is not a first, second, or third priority target for any of the sighter's weapons, infrequent intervisibility determinations are recommended (1.0).

M_6 = 1.0 if target is NOT first, second or third priority; 0.0 otherwise

5.4.4.2.3. Target damage status

Damaged targets are given less attention than healthy targets. If the target is destroyed or has no firepower, it is ignored (1.0). If it lacks mobility, it receives reduced attention (0.5), otherwise it receives maximum attention (0.0).

M_7 = 1.0 if target has firepower kill or is destroyed; 0.5 if it has mobility kill; 0.0 otherwise

5.4.4.2.4. Proximity to target

Entities check more often when an enemy entity is relatively close. Close is computed in terms of the target's firing range, and the amount of attention paid is proportional to its distance (zero yields 0.0, a distance exactly equal to the target's firing range yields 1.0).

$$M_8 = \frac{d}{r}$$

where:

d is the distance to the target entity
r is the maximum range of any weapon possessed by the target entity

This behavior is similar to the case in the sighter-based heuristics where a sighter varied its intervisibility update rate to a target depending on their relative distance. However, in one case the sighter's maximum weapons' range is used to determine its intervisibility update rate while in the other, the target's maximum weapons' range is used.

In Figure 5.4.4.2.4-A the circles represent the maximum range of weapons of entities A and B. Consider the sighter-based sub-heuristic with A being the sighter entity and B the target entity. A's intervisibility update rate should not increase because B is beyond its (the sighter's) maximum range of weapons. However, if B is the sighter entity, B's intervisibility update rate should increase.

In contrast, consider the target-based sub-heuristic with A being the sighter entity and B being the target entity. A's intervisibility update rate should increase. When B is the sighter entity, B's intervisibility update rate should not increase.

Table 5.4.4.2.4-A summarizes the difference:

Table 5.4.4.2.4-A Effect of sub-heuristic on sighter's intervisibility update rate

	Type of sub-heuristic	
Sighter	Sighter-based	Target-based
A	No increase	Increase
B	Increase	No increase

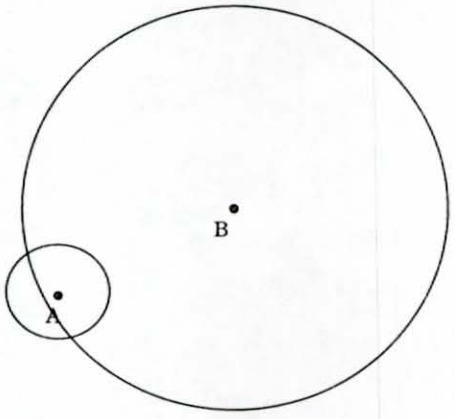


Figure 5.4.4.2.4-A Sighter and target maximum weapons' ranges

In these experiments, the vehicle's maximum weapons' ranges were identical. For simplicity, the target's maximum weapons' range was used for both sighter and target based sub-heuristics.

6. Evaluation of the intervisibility heuristics

Evaluation of the heuristics requires:

- Establishing the performance metrics,
- Data collection, and
- Heuristic ranking based on effectiveness.

This section addresses these points in detail.

6.1. Evaluation experiment

The performance of the intervisibility heuristics was evaluated in the context of a set of three standard military scenarios, to be described later.

The performance of the CGF Testbed Simulator in the area of intervisibility was measured and compared for heuristic and "base" versions in each scenario. The "base" versions did not incorporate any heuristics; i.e., they were "no-heuristic" versions. Intervisibility performance was considered in terms of number of intervisibility determinations performed, sighting event times, and heuristic computational overhead.

A "sighting event" or "sighting" refers to the event of an unblocked line of sight being detected by an intervisibility determination for two entities for which there was no such line of sight just prior to the event. For example, a sighting event would occur when an entity crested a ridge and became visible to a hostile entity on the other side of the ridge and that fact was identified by an intervisibility determination between those two entities. The "sighting event time" is the simulation time of such an event.

This section discusses the performance metrics and the process of data gathering for the evaluation.

6.1.1. Performance metrics

The following data was gathered:

- The total number of sighting events,
- The time at which each sighting event occurred, the IDs of the sighter and target entities, and their locations, and
- The time, measured in clock ticks, at various stages of the intervisibility checking process.

The set of sighting events and sighting event times found by the no-heuristic version for a particular scenario is taken to be the "true" or "correct" set. When a heuristic is judged against the no-heuristic version the following cases arise:

- Sighting events may be *missed* by a heuristic
- There may be *extra* sighting events in the heuristic output
- Sighting events may be *delayed*
- Sighting events may occur *earlier*

Some sightings will be missed or be extra because of sampling error. Using coarse BUDs (1/2 second or more) in the test scenarios makes it inevitable that some transitions will be missed, both by the heuristic version (labeled "missed") and by the no-heuristic version (labeled

"extra"). The real question is how many sightings are missed because of delayed checking. Extra sightings are always from sampling error since the heuristics never do more frequent checks than the no-heuristic version of the system. It seems likely (and experiments support this) that the greater the average sighting delay for the System Under Test (SUT) the more misses will be recorded.

It is possible that the computational cost of computing some heuristic may exceed the cost of doing a real intervisibility determination. Both the sighter and target-based heuristics are quite complex, particularly when all their components are active. It is, therefore, not enough to simply count the number of intervisibility determinations done, rather, a comprehensive evaluation is needed that takes the computational cost of the heuristics in account.

Because the fine-grain heuristics produce a large number of internal messages, heuristics of this type must consider the message delivery overhead. With a sufficient *RATE* multiplier, the Simulator will spend most of its time delivering messages. It is necessary to account for this time both to determine an optimal multiplier and to evaluate the overall results.

To accomplish this, each time a clock interrupt occurs, the interrupt handler determines whether intervisibility code is active and, if so, at what level (message delivery, heuristic processing, or in the actual intervisibility computations). The results are formatted to indicate how much time was spent on the overall intervisibility calculations and on the individual parts of the intervisibility process.

It was found that message delivery time was a minor issue. Measurements yield delivery time on the order of 1/4 to 1/2 milliseconds (4000-2000 deliveries/sec.). As the number of targets rises the issue becomes even less important. (If there are no targets, all the intervisibility time is in overhead, and delivery time may be important).

On the other hand, overhead for heuristics is proportional to the number of targets. If l is the number of local vehicles and m is the number of remote vehicles, the heuristic overhead is proportional to their product $l*m$, but message delivery is proportional to l only.

With this in mind, the sighting delays and the computational overhead of a heuristic are used in measuring the "cost" of the heuristic.

6.1.1.1. Savings calculation

Naively, it may seem that the effectiveness of using a heuristic is the difference between the number of point-to-point intervisibility determinations expected by a user for the scenario with a particular setting of the BUD and the number of point-to-point intervisibility determinations actually done by a scenario. This is not true because the savings obtained may be optimistic as the heuristic overhead has not been taken into account. If the overhead to use the heuristic is high, the savings from a reduction in the number of intervisibility determinations may be offset. It is the net savings that must be used to evaluate the quality of a heuristic, not just the reduction in intervisibility computations.

The effectiveness of a heuristic is represented as the ratio of its savings to the cost the system has to incur to use it. Thus, we have:

$$E_{h,s} = \frac{S_{h,s}}{C_{h,s}}$$

where:

$E_{h,s}$ is the effectiveness of heuristic h for scenario s

$S_{h,s}$ is the savings in the number of intervisibility determinations achieved by heuristic h for scenario s

$C_{h,s}$ is the cost of using heuristic h for scenario s.

Table 6.1.1.1-A tabulates the data used to compute the *Overhead Multiplier* (OM). The OM for a heuristic is a measure of the overhead associated with using that heuristic. It is defined as the ratio of the total time spent processing point-to-point intervisibility determinations between vehicles with the heuristic to the total time spent in doing these determinations without a heuristic. The OM for each heuristic was computed by using a scenario consisting of 6 Blue and 6 Red circling vehicles so as to generate the maximum number of intervisibility checks. The invariable nature of the OM for each heuristic eliminates the need for computing it for each scenario (to be discussed later).

Table 6.1.1.1-A Computing the overhead multiplier.

Heur	Raw	Tics	t_{raw}	OM
Sym	266888	5223	4949	1.06
His	281911	5277	5227	1.01
Sgt	278664	5298	5167	1.03
Trg	269185	5255	4991	1.05
Fgh	280370	5284	5202	1.02
Fgs	283475	5284	5256	1.005
Fgt	276631	5288	5132	1.03

where:

Heur Name of the heuristic (refer to Section 5)

Raw Number of point-to-point intervisibility determinations done

Tics Total time spent by the SUT exercising intervisibility code (includes all heuristic overhead)

t_{raw} Time that would be taken if no heuristic was being used to do the checks (t_{raw} = Raw / 53.9)

OM Overhead multiplier (OM = Tics / t_{raw})

The denominator 53.9 used in computing t_{raw} is the number of point-to-point intervisibility determinations done per clock tick without using any heuristic. It is used to give an estimate of the time it would take, in *clock ticks*, to do the same number of intervisibility checks as that done by a heuristic. The clock tick is a measure of the granularity of the system clock. The clock in the personal computer used for the CGF Simulator ticks every 0.054945 seconds.

After the overhead multiplier has been determined, the savings $S_{h,s}$ can be determined.

$$S_{h,s} = (1 - (\eta * O_h)) * 100.0$$

where η is the number of sightings that heuristic h does in scenario s and O_h is the OM for heuristic h.

Example:

Assume the "Trg" heuristic does only 75% of the intervisibility determinations done by the no-heuristic version in the same scenario, thus showing an apparent saving of 25%. However, if it has an overhead multiplier of 1.05 the real saving is:

$$Sh,s = 1 - (0.75 \times 1.05) = 0.2125 \text{ or } 21.25\%$$

6.1.1.2. Missed and extra sightings calculation

It may seem that a heuristic should be penalized for "missed" sighting events. A sighting event is considered "missed" by a heuristic if it failed to produce a sighting event found by the no-heuristic version. It may be argued that a heuristic must be "bad" if it misses many sighting events (and "good" if it does not). However, the situation is more complicated than that.

When a scenario is repeated results from the second run are not generally the same as in the first run. Small system perturbations from various non-deterministic events, such as network packet delivery times, have cumulative effects resulting in different intervisibility determination sampling (although the frequency is the same). For a BUD of 1.0, if an entity becomes visible for less than a second, any given run may miss the event while another might see it. This at first may seem unlikely but the scenarios often offer opposing vehicles separated by kilometers of busy terrain. Such conditions often yield very short periods of visibility.

It was found that a no-heuristic simulator run against the test scenarios showed approximately 5% variability from run to run in terms of missed and extra sightings. For example, if two runs are made with the identical no-heuristic version and compared, the second will typically show about 5 extra signings and 5 missed sightings for each 200 sighting events in the run. In this case, the labels are absolutely correct: sightings labeled "extra" are sightings found in the second run and missed in the first run; sightings labeled "missed" are those found in the first run but not found for the second run. The values are usually about equal because neither run has any advantage; ultimately both figures represent sightings missed by test runs.

It would be difficult to directly reflect the missed events in the computation of the heuristic's cost. One real difficulty to overcome, beyond the variability problem, was how to allow for the misses generated by sighter and target-based heuristics which intentionally "missed" many sightings (such as from or to destroyed vehicles). Misses, other than those caused through sampling variation, are closely tied to the mean and standard deviation for the sighting delays. Heuristics that delayed sightings greatly are prone to miss sightings.

Analysis of the results of the combat scenarios (see Section 6.1.2), excluding the sighter and target data, revealed a positive correlation between the raw metric used for evaluation and the number of misses seen for the heuristic/scenario trial. The correlation was not very high (correlation coefficient was 0.374). The low correlation, the variability from run to run, and the difficulty of accounting for the misses led us to ignore this factor in evaluation.

6.1.1.3. Sighting delay calculation

The "sighting delay" is the difference in the simulation times of the same sighting event in the heuristic and no-heuristic version. The sighting delays must be a factor in determining a heuristic's cost (and hence its effectiveness). A heuristic should be penalized for delays; a heuristic that "sees" events earlier is preferable to another that "sees" them later. The absolute mean of the delays is used as one of the measures of the cost of using a heuristic.

One may argue that the mean of the signed delays should have been used instead of the absolute value, because if there are early sightings by a heuristic, it should be credited for it. However, the heuristics are not designed to sight earlier and most of the sighting's delays are positive. There may be some negative delays in sightings (i.e., sightings were done earlier) but these are generally due to sampling variations. A heuristic should not be given credit for sampling variations and so a signed value should not be used.

Another parameter used in the cost equation is the standard deviation of the absolute delays. If two heuristics have the same sightings delays, the heuristic having a smaller standard deviation is preferred over the other.

Although the cost of a heuristic should rise with the standard deviation, the standard deviation is deemed to be of less importance than the mean. To reduce the impact of the standard deviation its square root is used. The square root of a positive number is closer to 1 than the number itself, so using the square root of the standard deviation in the product rather than the standard deviation itself will result in the deviation contributing less to the metric. This has the net effect of requiring the standard deviation to quadruple to have the same impact as doubling the mean.

6.1.1.4. Heuristic cost calculation

Combining the cost parameters we obtain the following equation for the *raw measure* of the cost of using a heuristic h for scenario s.

$$R_{h,s} = \mu * \sqrt{\sigma}$$

where:

- $R_{h,s}$ is the raw measure of the cost of using heuristic h for scenario s
- μ is the absolute mean of the sighting delays induced by heuristic h for scenario s
- σ is the absolute standard deviation of the sighting delays

As mentioned earlier (Section 6.1.1.2), two runs of the same version of the Simulator will not do the same sightings; there is about a 5% run to run variation in missed events. Moreover, the sightings done in the two runs do not happen at the same time, even though the scenario unfolds the same time after time, thus causing sighting delays. The run to run variation is factored into the heuristic cost:

$$C_{h,s} = \frac{R_{h,s}}{R_{0,s}}$$

where:

- $C_{h,s}$ is the cost of using heuristic h for scenario s
- $R_{h,s}$ is the raw measure of the cost of using heuristic h for scenario s and

$R0,s$ is the nominal cost (variation) from run to run and is calculated from two runs of the no heuristic version

After the cost of using a heuristic Ch,s is determined, the effectiveness Eh,s is computed by dividing the savings Sh,s by the heuristic cost.

$$E_{h,s} = \frac{S_{h,s}}{C_{h,s}}$$

Section 6.2 details the experimental results.

6.1.2. Experimental design

Three types of engagements were used to test the heuristics:

- Meeting
- Delay
- Assault

Six scenarios were developed; a *movement-only* and a *combat version* for each type of engagement.

The experiment consisted of running the unmodified and several versions of the heuristically modified Simulator with each of these scenarios and collecting data for analysis.

Recall that a sighting event is the transition between no intervisibility and intervisibility. For example, at the moment an entity comes from behind a treeline so a second vehicle can see it, two sighting events occur (1 for each vehicle). Data is collected as sighting information whenever the intervisibility between vehicles transitions from *blocked to unblocked*.

6.1.3. Test scenarios

The scenarios use a wide variety of terrain. Rough terrain (undulating with many sight blocking features) was used to increase the opportunities for sighting events; such events are the basis for the heuristic's evaluations. The scenarios are described in Sections 5.1.3.1 through 5.1.3.6, and the script files for the scenarios are given in Appendix A.7. All scenarios use the standard SIMNET Fort Knox terrain data base.

6.1.3.1. Meeting scenario

In this scenario, three opposing force (OPFOR or Red) platoons and three US (Blue) platoons converge at the bridge located at (40500,40000). The opposing and US forces are both using the same plan of attack: they send two platoons of infantry fighting vehicles to flank the other force, while covering the advance of the tanks and trucks to the bridge.

For detailed information about each platoon see Table 6.1.3.1-A.

Table 6.1.3.1-A Meeting scenario.

Platoon	Starting Location	Final Location	Platoon strength	Platoon Formation
US 1	(40080, 42591)	(40032, 40419)	4 : M2's	Echelon
US 2	(40455, 42631)	(40495, 40332)	2 : M1's 2 : M975's	Wedge
US 3	(41021, 42550)	(40902, 40649)	4 : M2's	Echelon
OPFOR 1	(39080, 37540)	(40132, 40100)	4 : BMP's	Echelon
OPFOR 2	(40212, 37495)	(40359, 40056)	2 : T72's 2 : URAL 375F's	Wedge
OPFOR 3	(41169, 37589)	(41210, 40325)	4 : BMP's	Echelon

Initial positions

The US forces start north of the bridge at (40500, 40000), while the OPFOR are positioned south of the bridge (see Figure 6.1.3.1-A). The OPFOR has two platoons in echelon formation positioned to the right and left of the center platoon in wedge formation. The US forces have two platoons, one in column formation and one in echelon formation, to the left and right of the center platoon, which is in wedge formation.

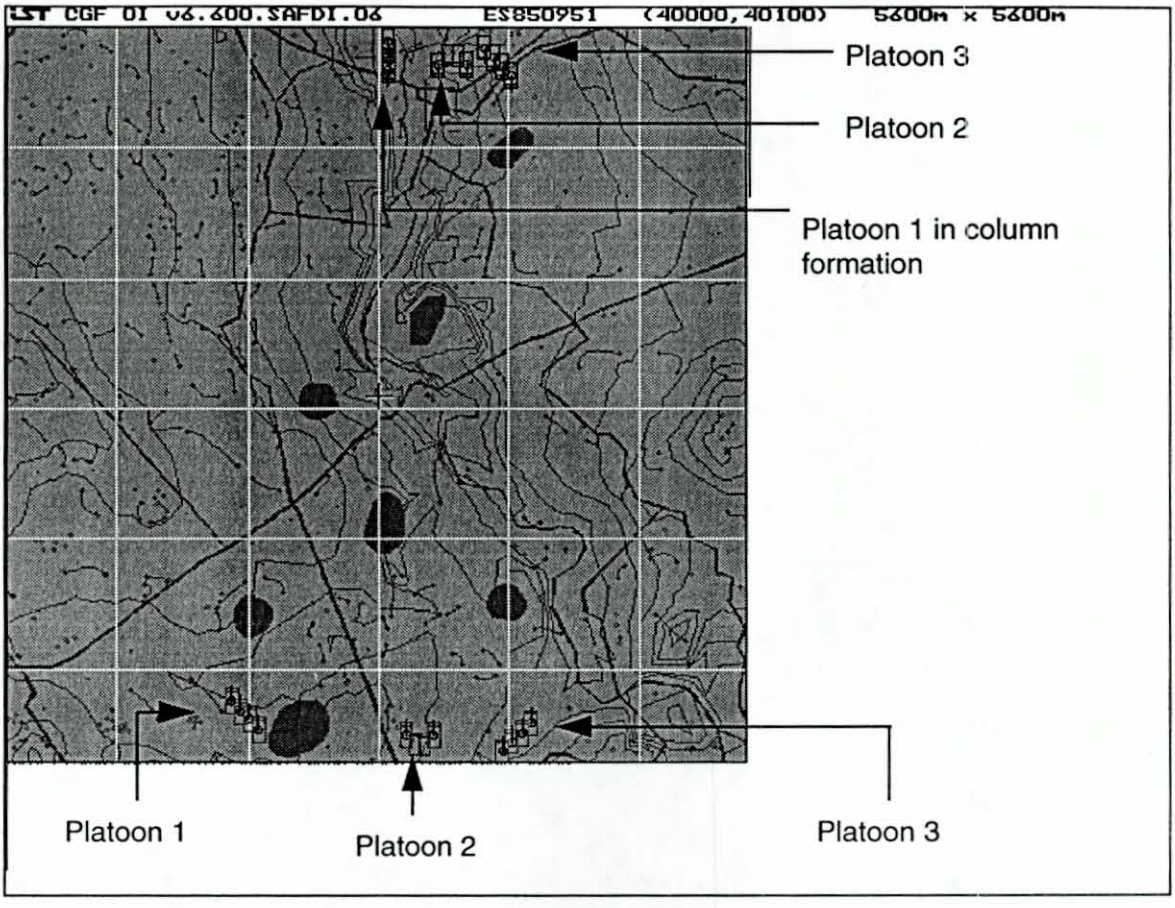


Figure 6.1.3.1-A Initial positions.

Movement routes

The forces have the same overall strategy, but their movements during the scenario are quite different since the OPFOR chooses a circuitous route while the US forces choose a more direct approach to the bridge (Figure 6.1.3.1-B). The OPFOR maneuver around obstacles as they head north to their destinations. OPFOR Platoons 2 and 3 break formation to avoid entering the trees, so as to increase visibility. The forces reform once the visual obstacles are cleared. The US forces take a more direct approach. US Platoon 1 forms a column and avoids the steep terrain to the east and the river to the west. This formation is held throughout until the river is forded near the objective. US Platoon 2 heads directly southward and US Platoon 3 swings wide (but always in the southerly direction) to avoid colliding with US Platoon 2.

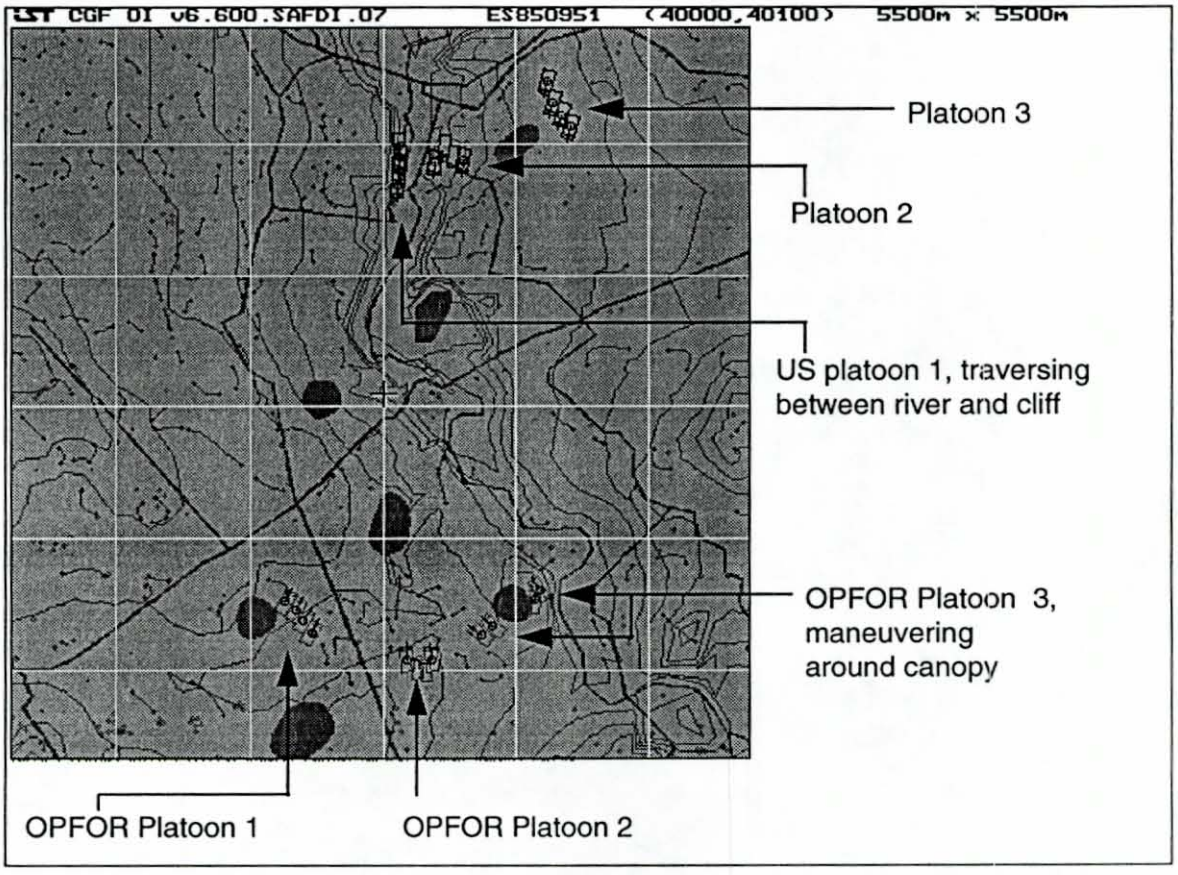


Figure 6.1.3.1-B Movements of both forces.

Final positions

As the scenario draws to a close, OPFOR Platoon 2 avoids a terrain hazard. To do this, they change their wedge into an echelon. As they pass the hazard, they resume their wedge formation until they reach their final position at the south end of the bridge. OPFOR Platoon 1 and US Platoon 3 both take a final position at the top of steep terrain to achieve an optimal offensive position. OPFOR Platoon 3, having crossed the river, now continues north until reaching its final position. US Platoon 1, having also crossed the river, reaches their final position in the valley. Finally, US Platoon 2 achieves its final position at the north foot of the bridge (Figure 6.1.3.1-C).

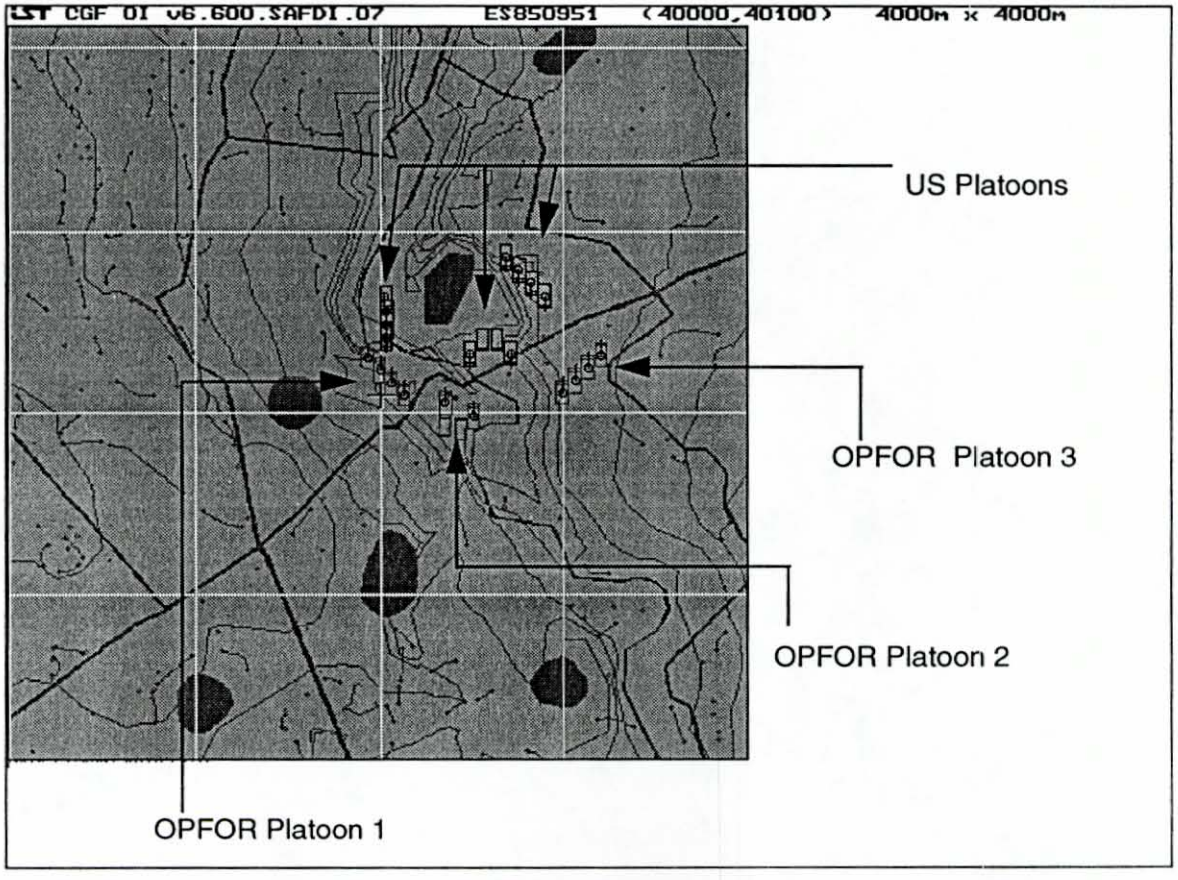


Figure 6.1.3.1-C Final Positions.

The combat version of the meeting scenario is the same as the standard meeting scenario except all entities have permission to fire at the enemy. This permission extends to the maximum range of their weapons (up to 4000 meters).

As soon as both forces are within range, they begin firing TOW and spandrel missiles. Since the vehicles are just within weapons range, many of these missiles fall short of their intended target. When the vehicles are well within range of each other, enemies are destroyed rapidly.

6.1.3.2. Delay scenario

In this scenario, the OPFOR is trying to delay the US force, while the OPFOR "main force" escapes to the west. (The main force does not actually exist in the scenario). The US forces do not "know" they are being delayed and believe they are pursuing the main force.

For detailed information about each platoon see Table 6.1.3.2-A.

Table 6.1.3.2-A Delay scenario.

Platoon	Starting Location	Final Location	Platoon strength	Platoon Formation
US 1	(23383, 23885)	(21513, 22663)	4 : M2's	Echelon
US 2	(23879, 23141)	(20852, 22553)	2 : M1's 2 : M975's	Wedge
US 3	(23547, 21693)	(20008, 22828)	4 : M2's	Echelon
OPFOR 1	(21025, 24105)	(21159, 21573)	2 : BMP's 2 : T72's 2 : URAL 375F's	Box (With fuel trucks in the center.)

Initial Positions

The US forces start about 3000 meters to the east of the opposing force (Figure 6.1.3.2-A).

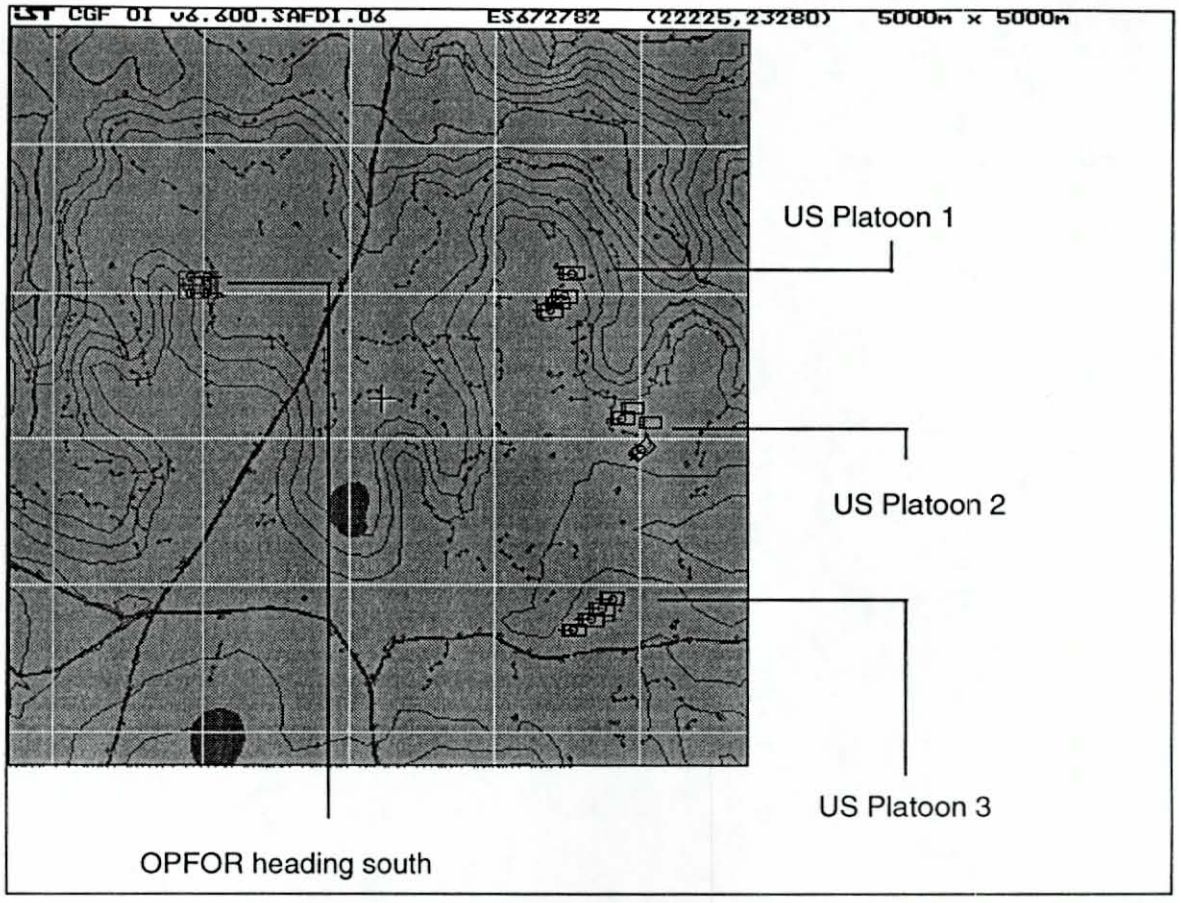


Figure 6.1.3.2-A Initial positions. OPFOR in box formation heads south.

Phase 1

The first phase of the OPFOR delay strategy is to head south to draw the attention of the US platoons. Since the OPFOR platoon's movement to the south is quite visible, all US platoons head south to prevent the OPFOR from escaping in that direction. The OPFOR moves south until it acquires a defensive position behind a group of trees, while the US forces continue to head south and then take defensive positions to wait for a possible OPFOR break to the east (see Figure 6.1.3.2-B).

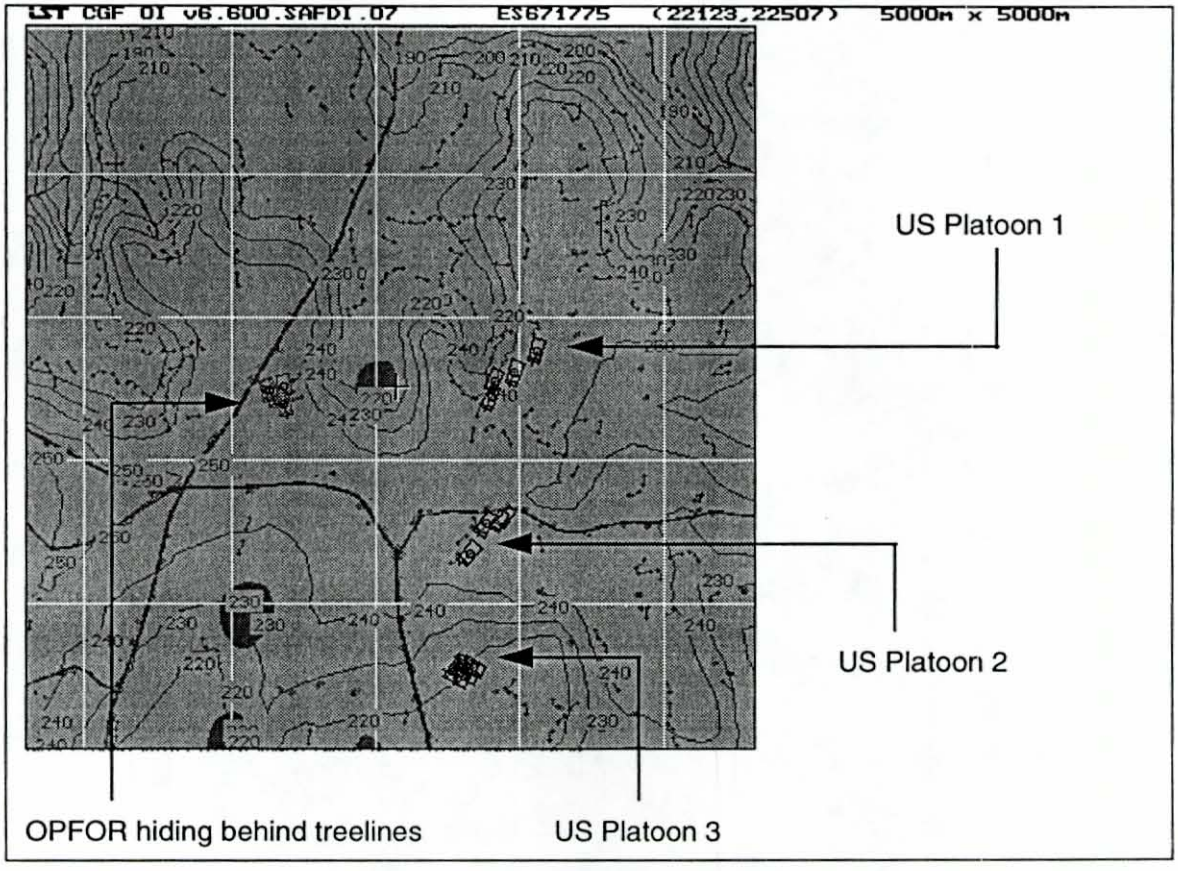


Figure 6.1.3.2-B OPFOR takes up a defensive position.

Final phase

Next, the US forces attack the vulnerable OPFOR. The main attack is lead by US Platoon 2. US Platoons 1 and 3 begin to flank the OPFOR to protect platoon 2 and to narrow the escape routes of the OPFOR platoon. When the US forces attack, the OPFOR retreats to the valley in the west. As they retreat, two T72s stay behind to cover the rest of the platoon's escape. After the main part of the OPFOR platoon is close to the edge of the valley, the T72s join in the retreat. The US forces continue their advance on the retreating OPFOR (Figure 6.1.3.2-C).

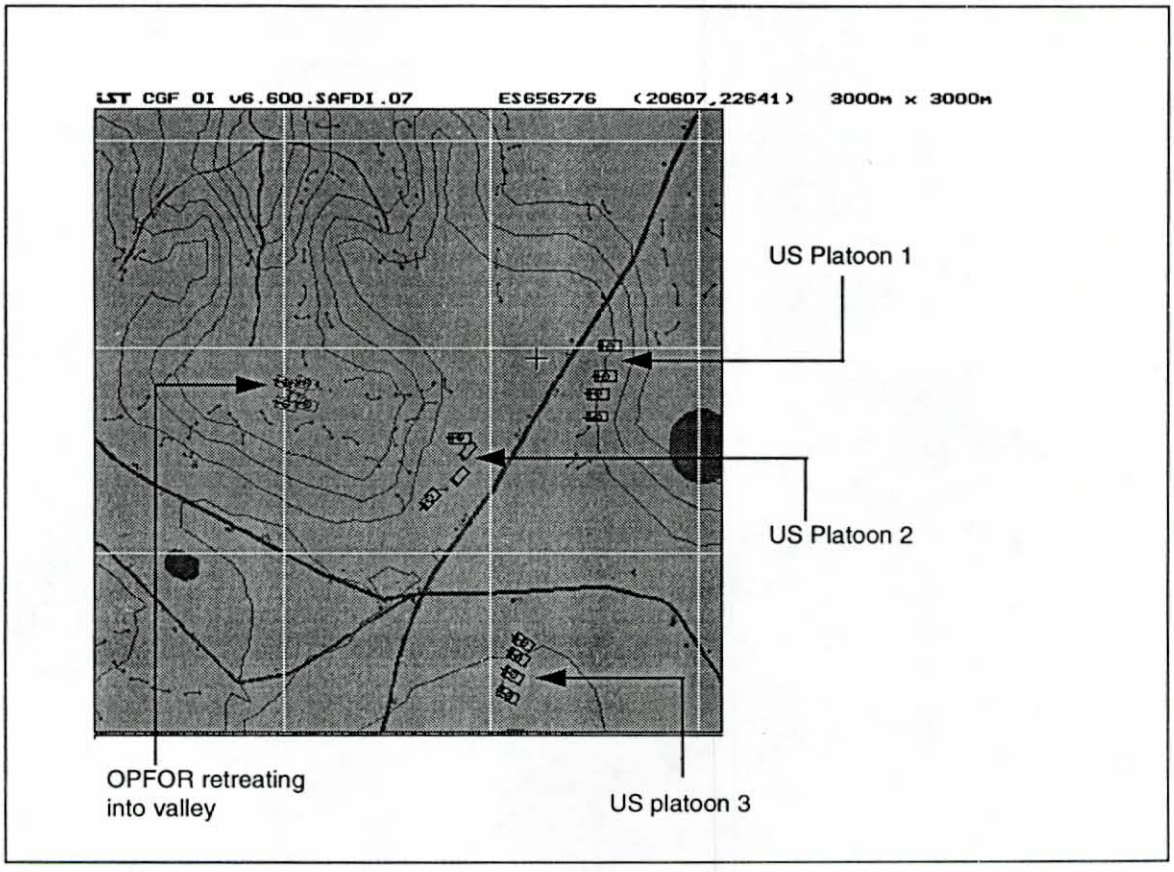


Figure 6.1.3.2-C The OPFOR retreats into the valley.

The combat version of the delay scenario is the same as the standard delay scenario except all entities have permission to fire at the enemy. This permission extends to the maximum range of their weapons (up to 4000 meters).

Since the vehicles are initially well within weapons range, the OPFOR capabilities are quickly destroyed by the US forces. Even though the US forces greatly outnumber the OPFOR, the OPFOR does a significant amount of damage to the US forces before it is destroyed.

6.1.3.3. Assault scenario

In this scenario, one OPFOR platoon will try to defend a bridge while three US platoons attempt to take the bridge.

For detailed information about each platoon see Table 6.1.3.3-A.

Table 6.1.3.3-A Assault scenario.

Platoon	Starting Location	Final Location	Platoon strength	Platoon Formation
US 1	(7370, 6593)	(3216, 8762)	4 : M2's	Echelon
US 2	(7493, 6703)	(3179, 9093)	2 : M1's 2 : M975's	Wedge
US 3	(7628, 6801)	(3142, 9620)	4 : M2's	Echelon
OPFOR 1	(3718, 9105)	Same	3 : BMP's 1 : T72's 2 : URAL 375F's	None

Initial positions

The US forces start southeast of the bridge at (3012, 9170), while the opposing forces are positioned east of the bridge, with the T72 directly at the foot of the bridge as a last defender. The rest of the forces (BMPs and trucks) are positioned to the east and southeast of the bridge (Figure 6.1.3.3-A).

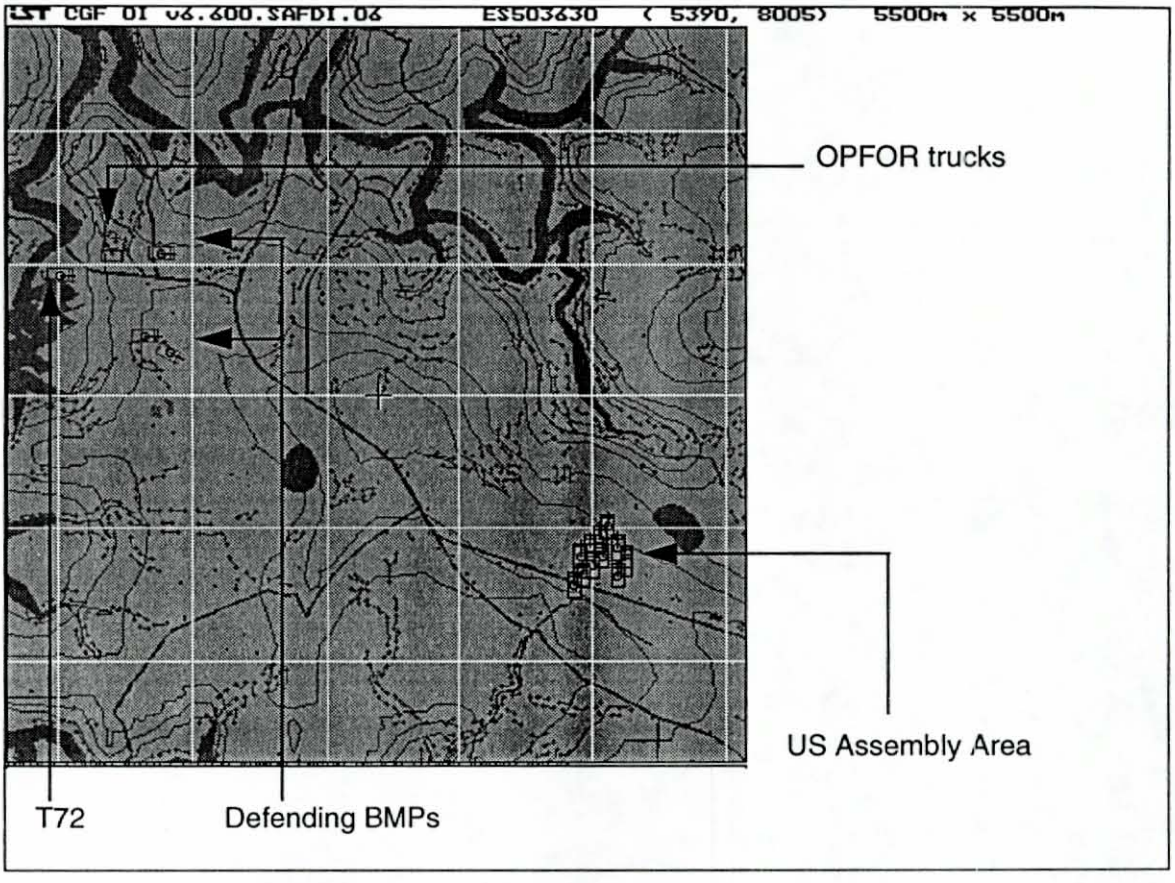


Figure 6.1.3.3-A Initial Positions of forces.

The assault

The US force, having assembled to the southeast of the bridge, breaks into the three platoons (Figure 6.1.3.3-B). US Platoon 1 swings west, moving along the tree lines, for a right flank attack on the opposing platoon. US Platoon 2 heads northwest for a direct assault on the bridge. US Platoon 3 swings northwest for a left flank attack on the opposing platoon. As US Platoons 1 and 2 emerge from the last tree lines before the objective, US Platoon 3 heads north at double speed. During the entire US forces' assault of the bridge, the opposing forces hold their positions.

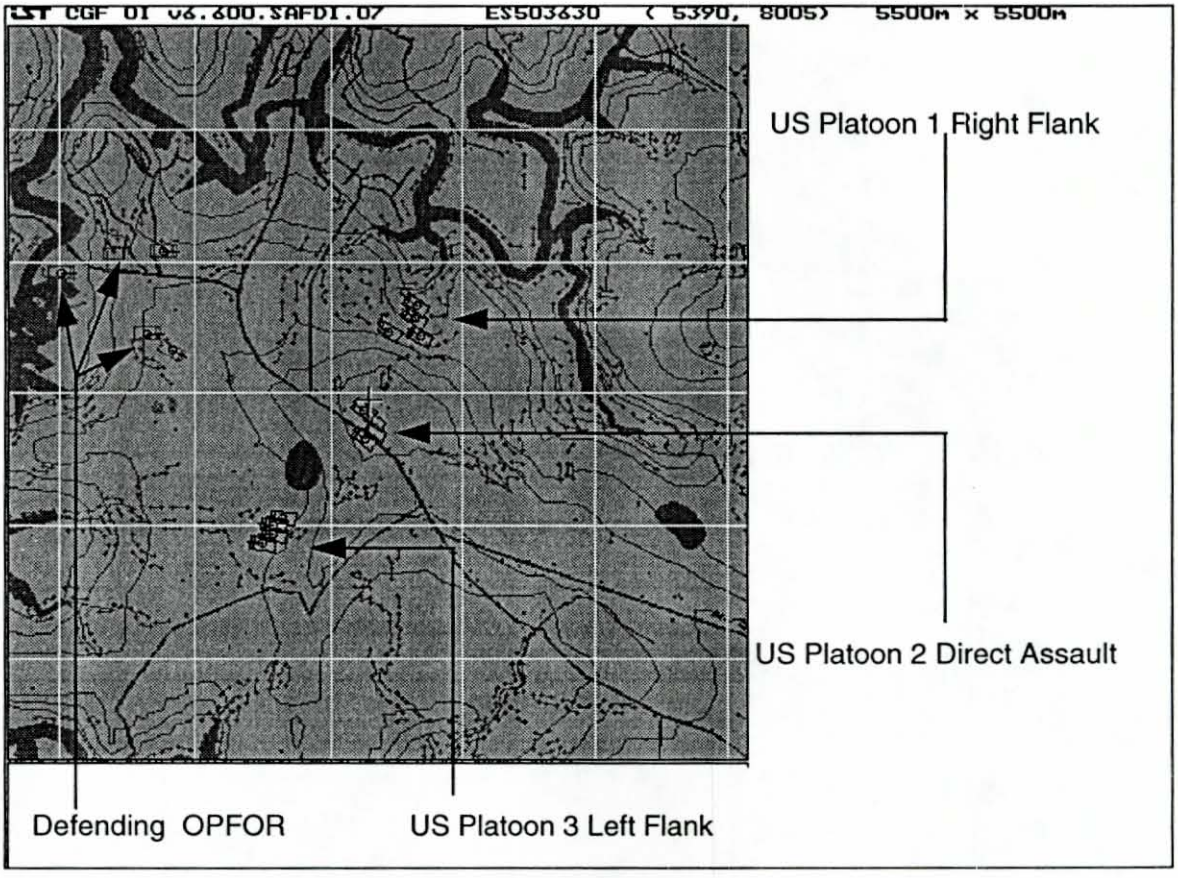


Figure 6.1.3.3-B Movements of US forces.

Final positions

This scenario ends when the US Platoon 2 and US Platoon 3 converge at the entrance of the bridge, while US Platoon 1, positioned on higher ground, is positioned north of the bridge (Figure 6.1.3.3-C).

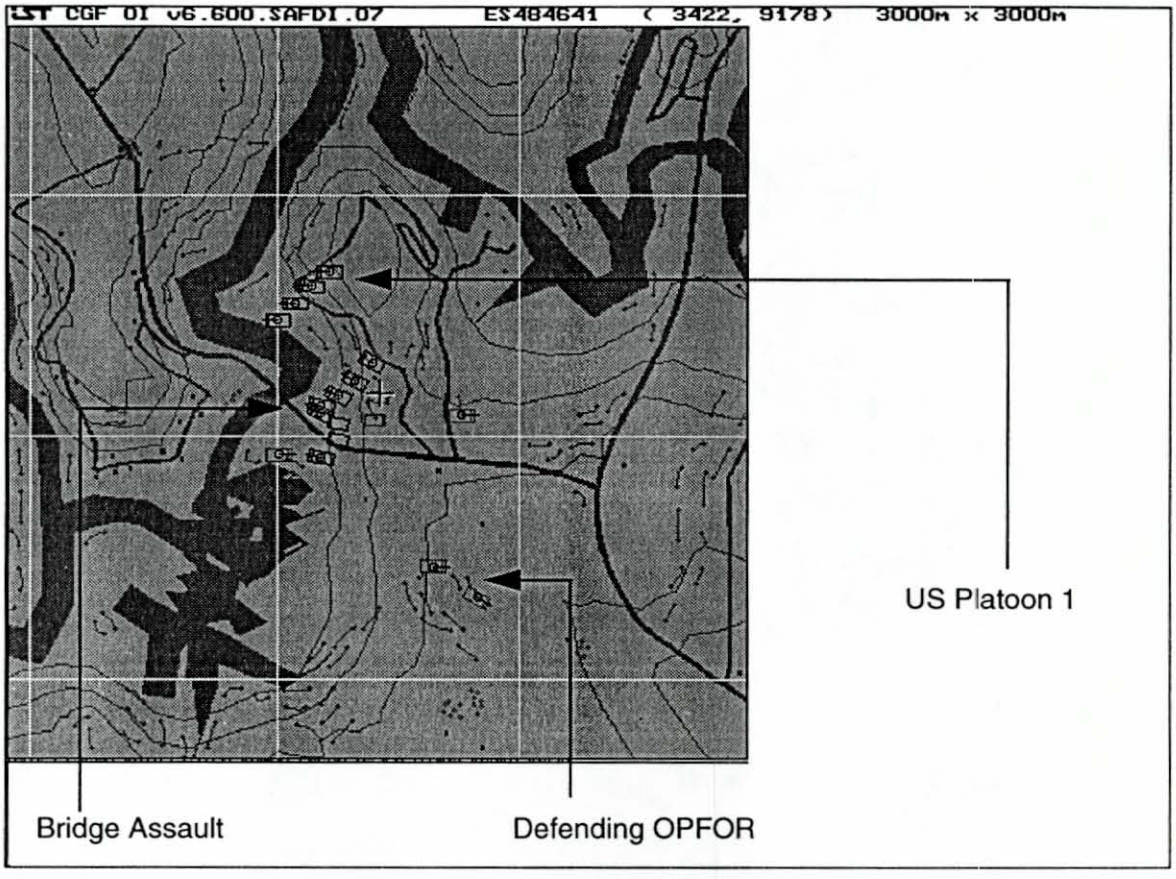


Figure 6.1.3.3-C Final assault on the bridge.

The combat version of the assault scenario is the same as the standard assault scenario except all entities have permission to fire at the enemy. This permission extends to the maximum range of their weapons (up to 4000 meters).

Since there is excellent cover for the US forces, combat does not ensue until the US forces are fairly close to the objective. Whenever intervisibility is achieved, the US forces are able to eliminate the BMP's positioned to the east and southeast of the bridge. However, because of its excellent position at the foot of the bridge, the OPFOR T72 does damage the offensive capabilities of the US forces as they begin their final assault of the bridge.

6.1.4. Data collection

A project of this complexity requires the analysis of large amounts of data. The scenarios that were developed for the evaluation of heuristics ran from 6 to 10 minutes (Section 6.1.3). For each scenario, data was collected for 7 heuristics. Two runs were made without heuristics; one became the reference data, and the other a "base version" used to evaluate run to run variability.

6.1.4.1. Scenario divergence

Initially, data was collected by creating a point-to-point network (to reduce network processing) between two Simulators; each running its part of the scenario. This approach did not prove viable. Because of the ill conditioned nature of the experiment (refer Appendix A.6) a second run of a scenario would usually diverge from the first. Hence message delivery would not follow the same order which indirectly caused vehicles to travel slightly perturbed routes, leading to ever more important differences between the runs. By the end of a run of more than a few minutes the sighting event histories would be very different.

It is impractical to compare the sighting data from two runs of the "same" scenario if the scenarios in question are different. In initial runs location variability of entities between runs were observed to vary by as much as 40 m or more.

6.1.4.2. Logging the scenarios

The problem of scenario divergence was solved by logging each scenario's network traffic. For the evaluation of a system, the logged data was re-played; the scenarios were re-created exactly in terms of the network activity. The logging process was automated to remove errors in synchronizing the start and end of the scenarios. Two personal computers ran the Simulators, while a third logged the network traffic. The entire arrangement was done on a three machine network, so there was no extraneous traffic.

To automate data collection, one machine played the role of the *Master* and the other the *Slave*. When the Slave was started, it waited until it received a "go" message from the Master.

Modifications were made to the Testbed to support the needed synchronization. The Slave is capable of giving itself a "wait" message with a "key." It is removed from the wait state only when a "wait complete" message arrives from the Master. The Slave verifies the key and, if it is correct, leaves the wait state. Also, a Master can cause a Slave to exit the simulation by sending an exit message.

6.1.4.3. Scenario playback for testing heuristics

After the scenarios had been logged, they were played back to generate test data for heuristic evaluation using a two PC point-to-point arrangement. One PC played the logged scenarios repeatedly, while the second PC ran a modified Simulator that did intervisibility tests between entities that were on remote machines.

Whenever appearance packets arrived a dead-reckoning model was updated (or created if it did not exist). Intervisibility checks, including heuristic processing, were done between the entities' locations as given by the dead reckoning models.

6.2. Experimental results

The following sections discuss the performance of the heuristics. The performance of a heuristic is based on its performance in different scenarios.

All the components of the sighter and target-based heuristics, both the fine and coarse-grain versions, were given equal weights so all of the components could be exercised (refer to Section 5.4.4.1 and Section 5.4.4.2 for a description of the sighter and target-based heuristics). It was not practical to examine permutations of the heuristic components and split points in the available time.

6.2.1. Heuristics' performance by scenario

Each of the following subsections discusses an implemented heuristic: history, symmetry, sighter and target-based heuristics.

The effect of varying the intervisibility base update rate (or BUR) will be examined in Section 6.2.1.1. That section shows the results for each BUR setting.

Table 6.2.1-A displays the data used to compute the run to run variance, an important factor in calculating a heuristic's cost (refer to Section 6.1.1.4 for a discussion on computing the heuristic cost). The $|\mu|$ and the $|\sigma|$ values are obtained by running the no-heuristic version twice and comparing the data gathered by the two runs.

Table 6.2.1-A Computing run to run variance.

Scenario	$\|\mu\|$	$\|\sigma\|$	$R_{0,s} = \|\mu\| * \sqrt{\|\sigma\|}$
Meeting	0.21	0.34	0.122
Delay	0.31	0.32	0.175
Assault	0.26	0.39	0.162
Meeting (C)	0.30	0.26	0.153
Delay (C)	0.31	0.34	0.181
Assault (C)	0.29	0.31	0.161

The $R0,s$ column indicates the run to run variance for each scenario. To compute the cost, Ch,s, for each heuristic/scenario pair, the raw measure of the cost Rh,s of that heuristic/scenario pair is divided by the $R0,s$ value for the scenario s (see Section 6.1.1.4).

In the following tables the headings are:

Scenario	The name of the scenario (refer to Appendix A.1) The (C) suffix refers to the combat version of the scenario.
$\|\mu\|$	The absolute mean of the sighting delays caused by a heuristic
$\|\sigma\|$	The absolute standard deviation of the sighting delays caused by a heuristic
Savings	The percentage of intervisibility checks saved
Rh,s	The raw measure of the cost of using a heuristic

$C_{h,s}$ The modified cost and is obtained by dividing $R_{h,s}$ by $R_{0,s}$ for the same scenario

$E_{h,s}$ The effectiveness of the heuristic

6.2.1.1. Varying the intervisibility base update rate

Varying the base update rate (BUR) means varying the frequency or rate at which intervisibility update messages are sent to each vehicle in the simulation. Strictly speaking varying the BUR does not fall into the category of heuristics; it was implemented to see what effects, if any, different BURs would have on the sighting events. Varying the BUR is insensitive to battlefield events. In this section the term "LX.X" stands for a simulation run with a BUD of X.X seconds between updates.

The statistics for the Simulator with a BUR of 2 (i.e., 2 updates/second) is given in Table 6.2.1.1-A and Figure 6.2.1.1-A.

Table 6.2.1.1-A L0.5 (BUR = 2) Effectiveness.

L0.5	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.39	0.26	-100.0	0.20	1.63	-61.3
Delay	0.38	0.26	-100.0	0.17	1.11	-90.3
Assault	0.35	0.24	-100.0	0.18	1.06	-94.4
Meeting (C)	0.43	0.28	-100.0	0.19	1.49	-67.2
Delay (C)	0.36	0.26	-100.0	0.23	1.01	-98.6
Assault (C)	0.34	0.25	-100.0	0.17	1.06	-94.7

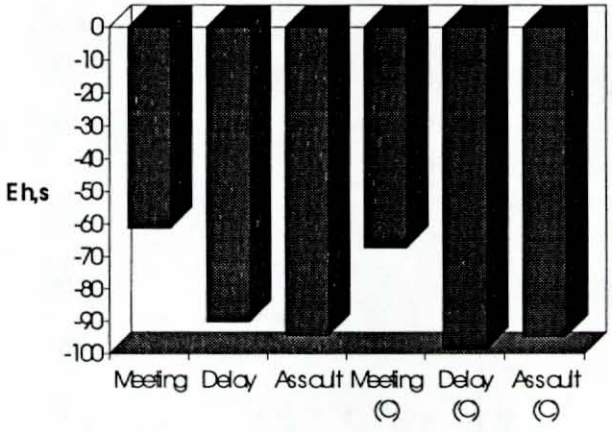


Figure 6.2.1.1-A L0.5 (BUR = 2) Effectiveness.

As can be seen, L0.5's effectiveness is uniformly negative because of the doubling of the BUR without a corresponding increase in the number of sightings.

The statistics of the data gathered by running the Simulator with a BUR of 0.67 (i.e., an update every 1.5 seconds) is shown in Table 6.2.1.1-B and Figure 6.2.1.1-B.

Table 6.2.1.1-B L1.5 (BUR = 0.67) Effectiveness.

L1.5	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.49	0.34	33.3	0.29	2.34	14.2
Delay	0.47	0.4	33.3	0.30	1.70	19.6
Assault	0.5	0.37	33.3	0.30	1.88	17.7
Meeting (C)	0.44	0.33	33.3	0.30	1.65	20.1
Delay (C)	0.49	0.38	33.3	0.25	1.67	19.9
Assault (C)	0.53	0.4	33.3	0.34	2.08	16.0

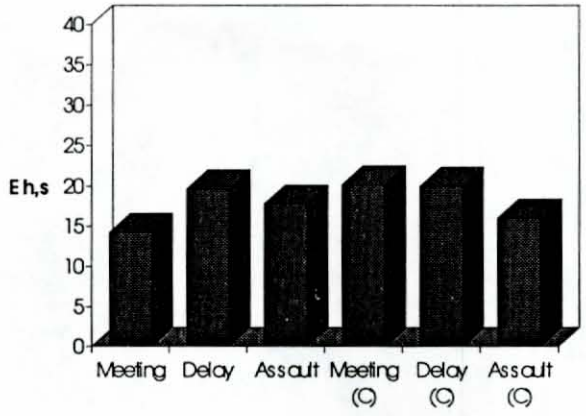


Figure 6.2.1.1-B L1.5 (BUR = 0.67) Effectiveness.

Setting the BUR to 0.67 updates/sec. results in positive effectiveness for all scenarios.

The statistics of the data gathered by running the Simulator with a BUR of 0.5 (an update every 2.0 seconds) is shown in Table 6.2.1.1-C and Figure 6.2.1.1-C.

Table 6.2.1.1-C L2.0 (BUR = 0.5) Effectiveness.

L2.0	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.64	0.47	50.0	0.44	3.60	13.9
Delay	0.66	0.52	50.0	0.49	2.72	18.3
Assault	0.68	0.52	50.0	0.46	3.03	16.5
Meeting (C)	0.61	0.47	50.0	0.48	2.73	18.2
Delay (C)	0.65	0.5	50.0	0.42	2.54	19.6
Assault (C)	0.7	0.5	50.0	0.49	3.07	16.2

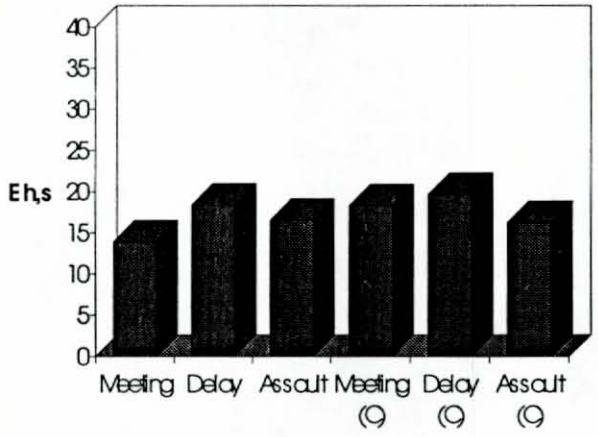


Figure 6.2.1.1-C L2.0 (BUR = 0.5) Effectiveness.

Running the Simulator with a BUR of 0.5 results in positive effectiveness similar to the effectiveness of a BUR of 0.67.

The statistics of the data gathered by running the Simulator with a BUR of 0.33 (an update every 3.0 seconds) is shown in Table 6.2.1.1-D and Figure 6.2.1.1-D.

Table 6.2.1.1-D L3.0 (BUR = 0.33) Effectiveness.

L3.0	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	1.06	0.77	66.6	0.93	7.62	8.7
Delay	1.07	0.79	66.6	0.97	5.43	12.2
Assault	1.07	0.82	66.6	1.02	5.98	11.1
Meeting (C)	1.14	0.76	66.6	0.95	6.50	10.2
Delay (C)	1.13	0.81	66.6	0.99	5.62	11.8
Assault (C)	1.11	0.82	66.6	1.01	6.24	10.6

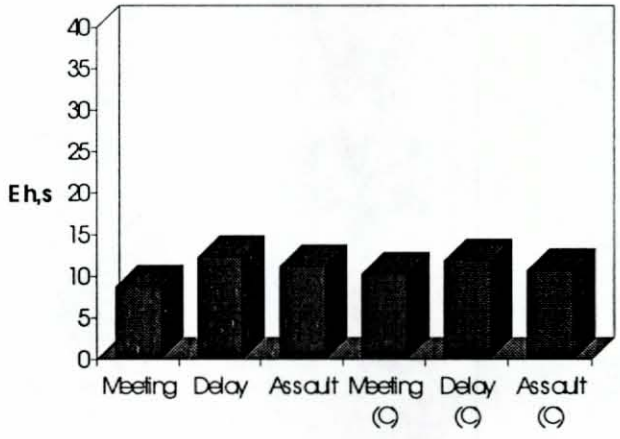


Figure 6.2.1.1-D L3.0 (BUR = 0.33) Effectiveness.

By comparing this data with the data for L2.0 it can be seen that L3.0 is not as effective as L2.0.

The statistics of the data gathered by running the Simulator with a BUR of 0.25 (an update every 4.0 seconds) is shown in Table 6.2.1.1-E and Figure 6.2.1.1-E.

Table 6.2.1.1-E L4.0 (BUR = 0.25) Effectiveness.

L4.0	$\|\mu\|$	$\|\sigma\|$	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	1.59	1.05	75.0	1.63	13.35	5.6
Delay	1.65	1.2	75.0	1.81	10.33	7.2
Assault	1.58	1.1	75.0	1.66	10.23	7.3
Meeting (C)	1.51	1	75.0	1.51	9.87	7.6
Delay (C)	1.65	1.25	75.0	1.84	10.19	7.3
Assault (C)	1.64	1.09	75.0	1.71	10.63	7.0

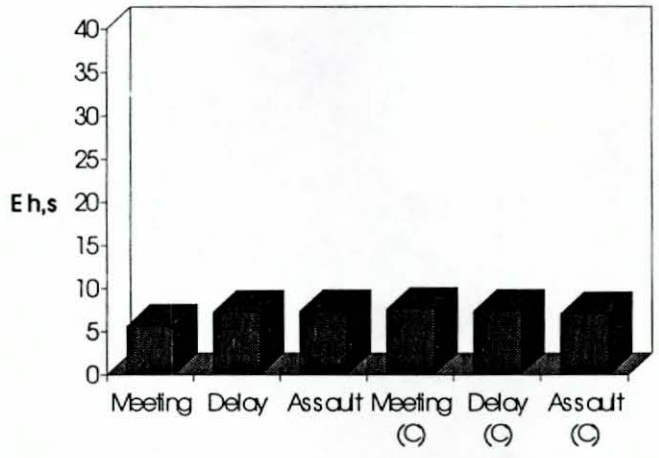


Figure 6.2.1.1-E L4.0 (BUR = 0.25) Effectiveness.

Relatively small, but positive, effectiveness values are seen.

The statistics of the data gathered by running the Simulator with a BUR of 0.20 (an update every 5.0 seconds) is shown in Table 6.2.1.1-F and Figure 6.2.1.1-F.

Table 6.2.1.1-F L5.0 (BUR = 0.20) Effectiveness.

L5.0	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	2.23	1.4	80.0	2.64	21.63	3.6
Delay	2.07	1.53	80.0	2.37	14.63	5.4
Assault	1.99	1.42	80.0	2.79	14.64	5.4
Meeting (C)	2.15	1.39	80.0	2.56	16.57	4.8
Delay (C)	2.23	1.57	80.0	2.53	15.44	5.1
Assault (C)	1.96	1.37	80.0	2.29	14.25	5.6

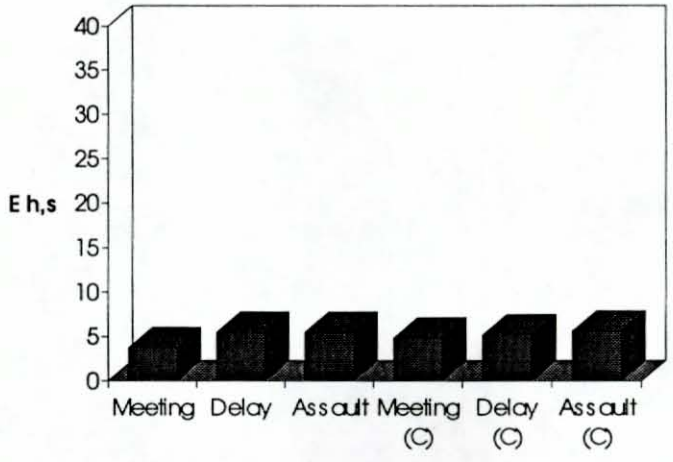


Figure 6.2.1.1-F L5.0 (BUR = 0.20) Effectiveness.

Small but positive effectiveness values are observed.

Accumulating the results of varying the BUR, we see:

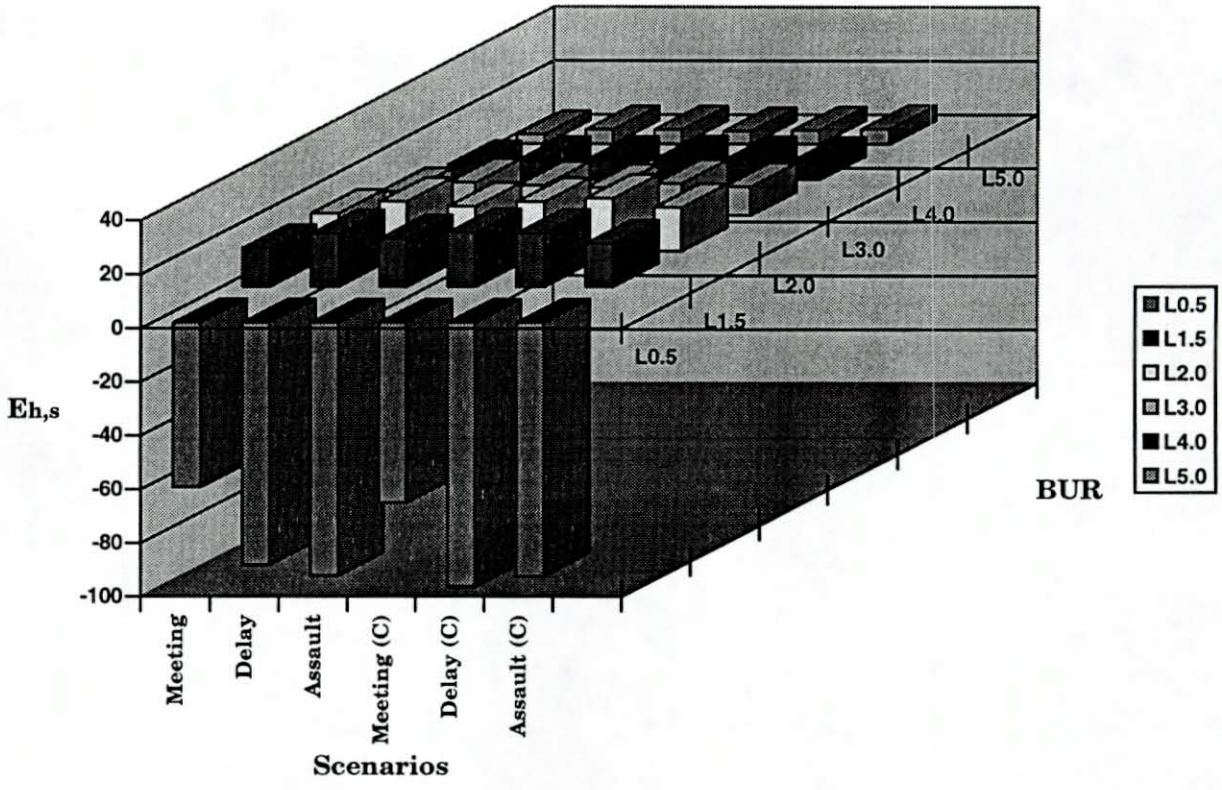


Figure 6.2.1.1-G Comparison of varying the BUR.

6.2.1.2. History heuristic

The statistics for the history (His) heuristic in shown in Table 6.2.1.2-A and Figure 6.2.1.2-A.

Table 6.2.1.2-A History effectiveness.

Scenario	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.57	0.45	45.6	0.382	3.13	14.6
Delay	0.49	0.48	40.7	0.339	1.94	21.0
Assault	0.55	0.49	46.5	0.385	2.38	19.5
Meeting (C)	0.66	0.48	47.1	0.457	2.99	15.8
Delay (C)	0.57	0.44	44.8	0.378	2.09	21.4
Assault (C)	0.52	0.53	47.1	0.379	2.35	20.0

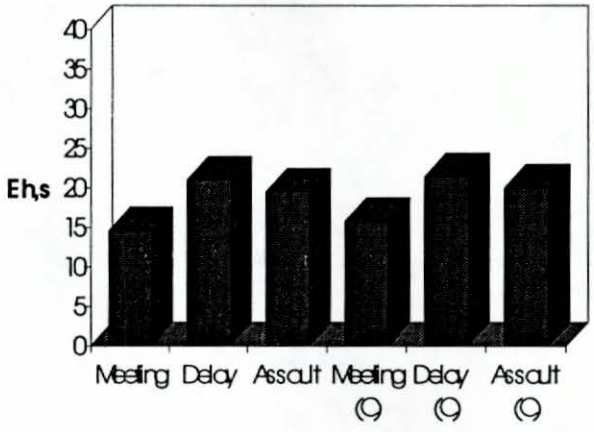


Figure 6.2.1.2-A History effectiveness.

A heuristic is "stable" if its range of Eh,s values from scenario to scenario is small. History is seen to be a stable heuristic. It is effective for all types of scenarios; combat or non-combat. The stability of the history heuristic is apparent by the uniformity of its Eh,s values.

6.2.1.3. Symmetry heuristic

The statistics of the symmetry (Sym) heuristic are shown in Table 6.2.1.3-A and Figure 6.2.1.3-A.

Table 6.2.1.3-A Symmetry effectiveness.

Scenario	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.40	0.28	46.7	0.212	1.74	26.8
Delay	0.36	0.33	46.0	0.207	1.18	39.0
Assault	0.32	0.41	46.2	0.205	1.27	36.4
Meeting (C)	0.40	0.23	46.6	0.192	1.25	37.3
Delay (C)	0.43	0.24	46.3	0.211	1.17	39.6
Assault (C)	0.33	0.37	46.3	0.201	1.25	37.0

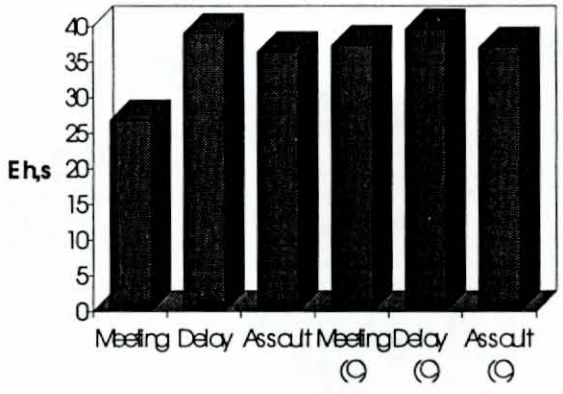


Figure 6.2.1.3-A Symmetry effectiveness.

As expected the results show symmetry to be very effective and stable. However, as previously mentioned, symmetry was only tested for entities simulated on the same network node. Applying symmetry across nodes would require network traffic to transmit the intervisibility determination results. That process was not tested.

6.2.1.4. Discrete sighter-based heuristic

The statistics of the coarse-grain sighter-based (Sgt) heuristic is shown in Table 6.2.1.4-A and Figure 6.2.1.4-A.

Table 6.2.1.4-A Discrete sighter-based effectiveness.

Scenario	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.19	0.31	0.708	0.106	0.869	0.815
Delay	0.28	0.30	-3.00	0.153	0.874	-3.43
Assault	0.26	0.36	-0.631	0.156	0.963	-0.655
Meeting (C)	0.25	0.29	20.7	0.135	0.882	23.5
Delay (C)	0.26	0.33	27.9	0.149	0.823	33.9
Assault (C)	0.26	0.31	19.1	0.145	0.901	21.2

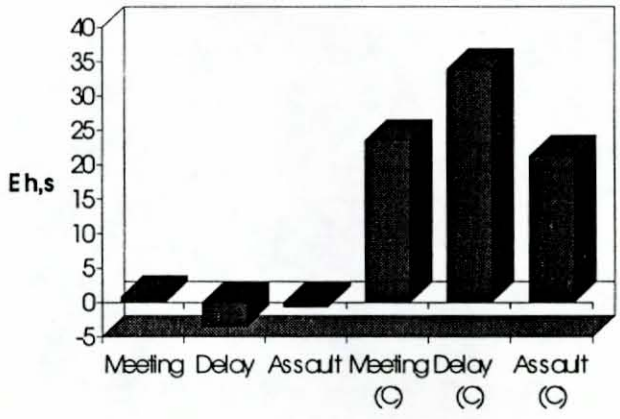


Figure 6.2.1.4-A Discrete sighter-based effectiveness.

The sighter-based heuristic performs better in combat situations than in non-combat situations. This is expected because this is where most of the sub-heuristics of the sighter-based heuristic come into play and save intervisibility checks. For example, intervisibility determinations will be skipped by a sighter if it has been destroyed and such a situation can arise only in combat.

For the non-combat delay and assault scenarios, the savings and $E_{h,s}$ are negative; the heuristic costs more and saves less. In such a situation it may be better *not to use the heuristic*.

6.2.1.5. Discrete target-based heuristic

The statistics of the coarse-grain target-based (Trg) heuristic is shown in Table 6.2.1.5-A and Figure 6.2.1.5-A.

Table 6.2.1.5-A Discrete target-based effectiveness.

Scenario	$\|\mu\|$	$\|\sigma\|$	Savings	Rh,s	Ch,s	Eh,s
Meeting	0.22	0.29	-0.380	0.118	0.967	-0.393
Delay	0.36	0.34	1.09	0.210	1.20	0.908
Assault	0.24	0.40	0.145	0.152	0.938	0.155
Meeting (C)	0.36	0.33	17.9	0.207	1.35	13.3
Delay (C)	0.39	0.38	39.8	0.240	1.33	29.9
Assault (C)	0.34	0.29	6.87	0.183	1.14	6.03

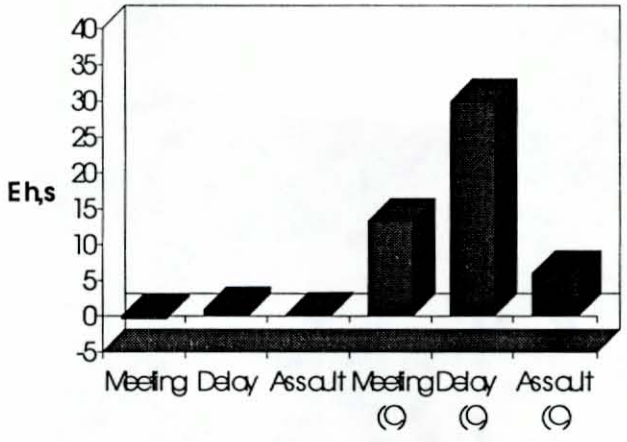


Figure 6.2.1.5-A Discrete target-based effectiveness.

Like the sighter-based heuristics, the target-based heuristic performs better in combat situations where its sub-heuristics are exercised. The sub-heuristics pertain to behaviors that are generally seen in combat situations. For example, intervisibility determinations to a target will be skipped if the target is destroyed and such a situation can arise only in combat. The heuristic is worse than no heuristic in the non-combat meeting scenario.

6.2.1.6. Continuous history heuristic

The statistics of the fine-grain history (Fgh) heuristic are shown in Table 6.2.1.6-A and Figure 6.2.1.6-B.

Table 6.2.1.6-A Continuous history effectiveness.

Scenario	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.46	0.35	31.4	0.272	2.23	14.1
Delay	0.39	0.30	25.4	0.214	1.22	20.8
Assault	0.44	0.35	36.5	0.260	1.60	22.8
Meeting (C)	0.54	0.41	36.8	0.346	2.26	16.3
Delay (C)	0.40	0.30	33.9	0.219	1.21	28.0
Assault (C)	0.44	0.36	38.8	0.264	1.64	23.7

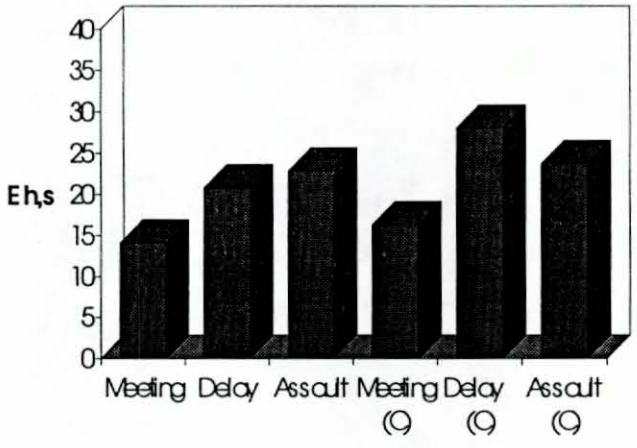


Figure 6.2.1.6-A Continuous history effectiveness.

Fine-grain history performs quite well for all scenarios, although its variability (maximum effectiveness - minimum effectiveness) is greater than the coarse-grain version's.

6.2.1.7. Continuous sighter-based heuristic

The statistics of the fine-grain sighter-based heuristic is shown in Table 6.2.1.7-A and Figure 6.2.1.7-A.

Table 6.2.1.7-A Continuous sighter-based effectiveness.

Scenario	$\|\mu\|$	$\|\sigma\|$	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.39	0.28	10.1	0.206	1.67	6.05
Delay	0.37	0.29	5.63	0.199	1.14	4.94
Assault	0.35	0.26	12.0	0.178	1.10	10.9
Meeting (C)	0.40	0.26	29.0	0.204	1.33	21.8
Delay (C)	0.37	0.27	36.4	0.192	1.06	34.3
Assault (C)	0.39	0.27	27.7	0.203	1.26	22.0

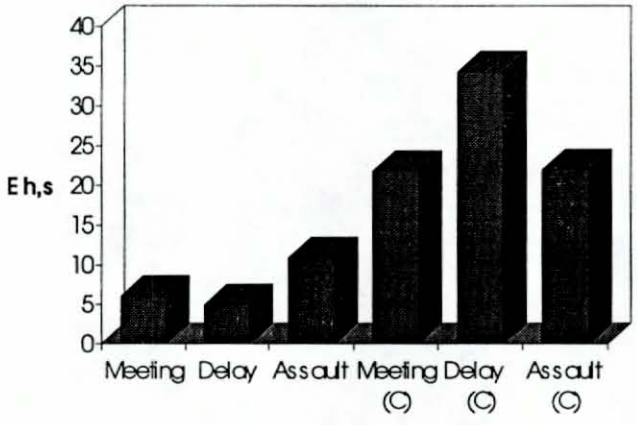


Figure 6.2.1.7-A Continuous sighter-based effectiveness.

The Fine-grain, sighter-based heuristic performs much better than the coarse-grain version - no negative *Eh,s* values were generated. The heuristic is more effective for combat scenarios than for the non-combat ones. This is because the sub-heuristics are exercised in combat situations leading to a saving in the number of intervisibility determinations.

6.2.1.8. Continuous target-based heuristic

The statistics of the fine-grain target-based heuristic is shown in Table 6.2.1.8-A and Figure 6.2.1.8-A.

Table 6.2.1.8-A Continuous target-based effectiveness.

Scenario	|μ|	|σ|	Savings	*Rh,s*	*Ch,s*	*Eh,s*
Meeting	0.38	0.28	8.33	0.201	1.65	5.05
Delay	0.42	0.32	11.3	0.238	1.36	8.31
Assault	0.38	0.28	9.57	0.201	1.24	7.72
Meeting (C)	0.38	0.29	33.0	0.205	1.34	24.6
Delay (C)	0.42	0.31	46.9	0.234	1.29	36.4
Assault (C)	0.37	0.27	29.4	0.192	1.19	24.7

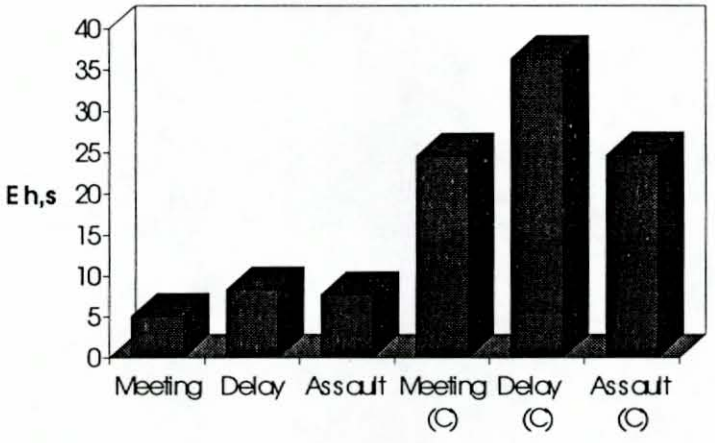


Figure 6.2.1.8-A Continuous target-based effectiveness.

Fine-grain target results closely parallel those for the fine-grain, sighter heuristics. The heuristic is more effective for combat scenarios. Again, this is because combat scenarios exercise the sub-heuristics leading to a saving in the number of intervisibility determinations.

Accumulating the results for the discrete (coarse-grain) heuristics we see:

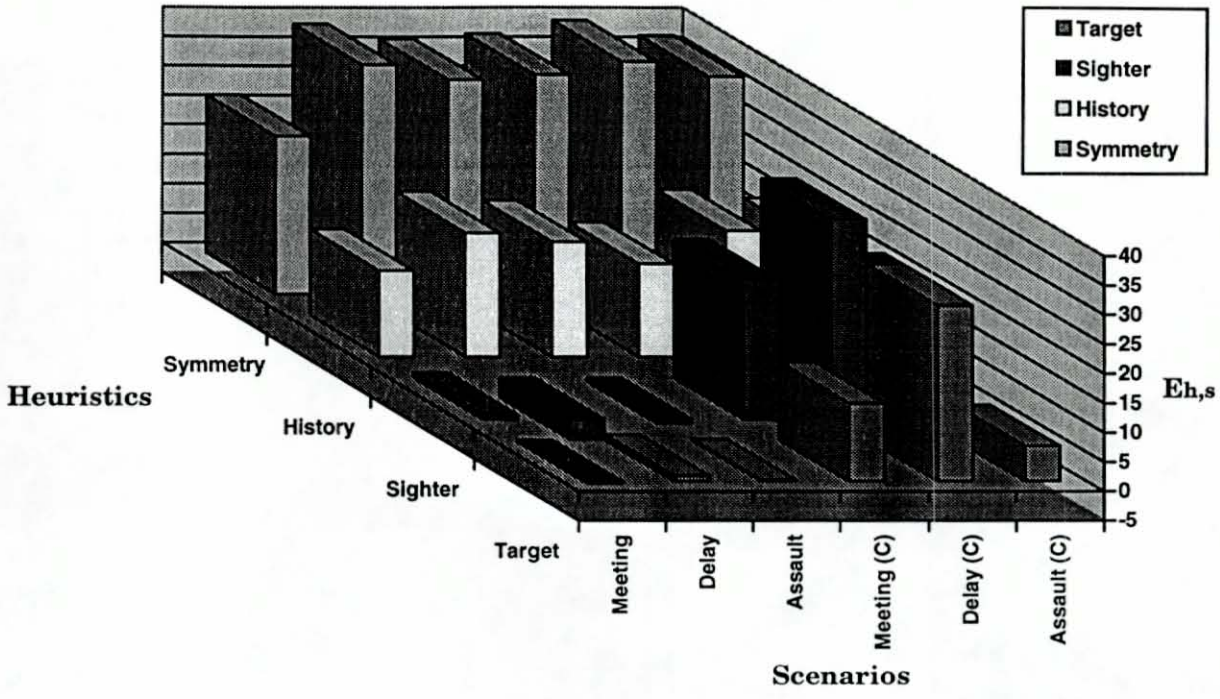


Figure 6.2.1.8-B Comparison of discrete (coarse-grain) heuristics.

Accumulating the result of the continuous (fine-grain) heuristics we see:

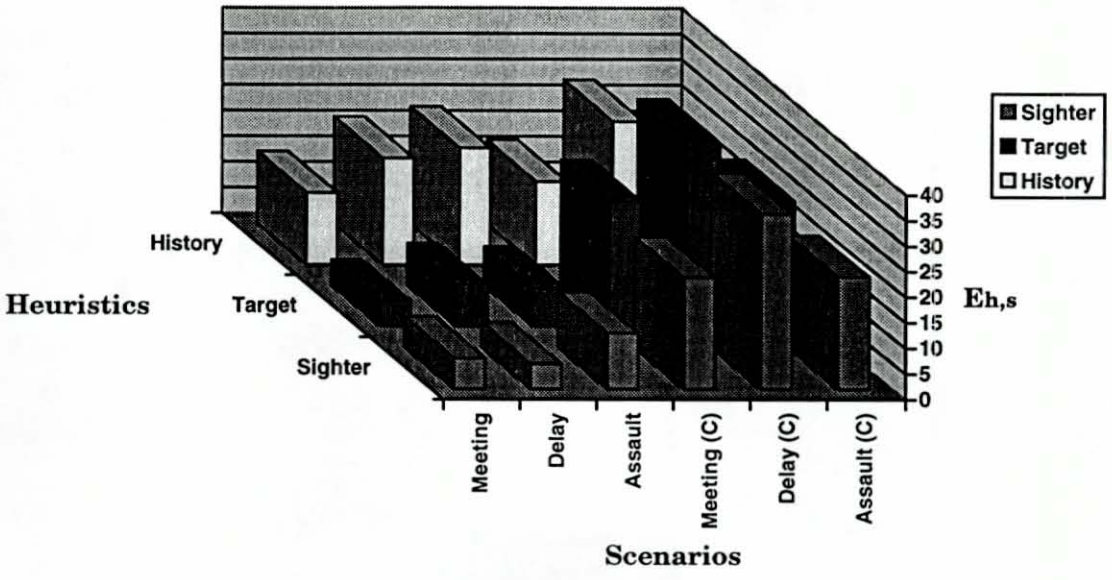


Figure 6.2.1.8-C Comparison of continuous (fine-grain) heuristics.

6.2.2. Overall heuristic performance

A heuristic's overall performance can be seen by comparing the range of its Eh,s values across heuristics. A good heuristic should have high Eh,s values and its variability should be small. Heuristic A is said to be more stable than B if its effectiveness is less dependent on the scenario used for evaluation.

Figure 6.2.2-A below shows the range of Eh,s values for the implemented heuristics, and Eh,s values for intervisibility BURs of 0.67 (a check every 1.5 seconds) and 0.5 (a check every 2 seconds). This figure shows the history heuristic to be the most stable heuristic. Sighter and target-based heuristics have very large spreads, with the extreme left points showing negative efficiency. The fine-grain sighter and target-based heuristics are marginally more stable and effective then their coarse-grain versions. The symmetry heuristic showed the greatest effectiveness with stability second to the history heuristic.

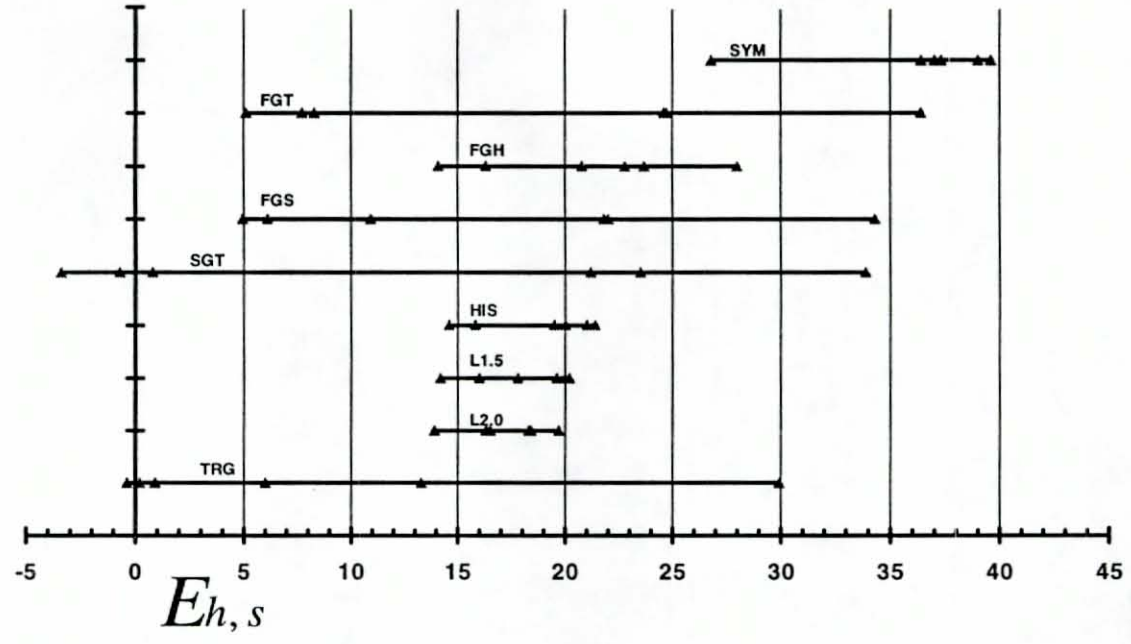


Figure 6.2.2-A Heuristic performance spread.

Interestingly, all heuristics performed well in vigorous situations. This is indicated by the appearance of the combat scenarios predominantly on the right of each line and the non-combat on the left.

To determine the overall effectiveness Eh of a heuristic a weighted average is used. Since heuristics have more to offer in complicated scenarios (the combat scenarios), more weight is given to such scenarios than to non-combat scenarios. Combat scenarios are weighted 3 times as much as non combat scenarios.

Eh can be computed as:

$$E_h = \frac{(E_{h,s1} + E_{h,s2} + E_{h,s3}) + 3(E_{h,s4} + E_{h,s5} + E_{h,s6})}{(E_{h,s1} + E_{h,s2} + E_{h,s3} + E_{h,s4} + E_{h,s5} + E_{h,s6})}$$

where:

Eh	is the overall effectiveness heuristic h
Eh,x	is the effectiveness of heuristic h for scenario x
$s1$	is the Meeting scenario
$s2$	is the Delay scenario
$s3$	is the Assault scenario
$s4$	is the Meeting (C) scenario
$s5$	is the Delay (C) scenario
$s6$	is the Assault (C) scenario

Using this metric the heuristics are "ranked" as shown in Table 6.2.2-A. The table also shows the overall savings, Sh, and the overall cost, Ch, of the heuristics using the same weights as were used for establishing the ranking. The heuristics are listed in descending order of overall effectiveness, i.e., from best to worst.

Table 6.2.2-A Heuristic rankings.

Heuristic	*Eh*	*Sh*	*Ch*
Symmetry (Sym)	37.0	46.4	1.3
Fine-grain target (Fgt)	23.2	29.7	1.30
Fine-grain history (Fgh)	22.0	35.1	1.7
Fine-grain sighter (Fgs)	21.3	25.5	1.2
Sighter (Sgt)	19.4	16.7	0.9
History (His)	18.9	45.8	2.5
L1.5*	18.3	33.3	1.8
L2.0*	17.6	50.0	2.8
Target (Trg)	12.4	16.2	1.2
L3.0*	10.9	66.6	6.2
L4.0*	7.2	75.0	10.5
L5.0*	5.1	80.0	15.8
L0.5*	-85.0	-100.0	1.2

* Refer to Appendix A.1 for an explanation of these symbols.

Data was gathered by running the Simulator with different BUD settings (50, 150, 200, 300, 400, and 500) to see the effects of the IUR on the sightings. These experiments are referred to by "LX.X" in Table 6.2.2-A. Table 6.2.2-A shows that except for L1.5 and L2.0, the BUD-based heuristics performed poorly and are at the bottom of the rankings. L1.5 and L2.0 seem to perform better than the target-based heuristic; but this is partly an illusion.

The target-based heuristic (as well as all other heuristics) save "intelligently." Intervisibility determinations are delayed only when they are deemed acceptable, for example, when a scenario is calm. However, no delays are allowed when the scenario becomes more active, for example, when combat starts. In the combat scenarios vehicles may be destroyed quite early in the scenario leading to the target-based heuristic delaying intervisibility determinations. In the non-combat versions intervisibility determinations may again be delayed due to the relative calmness of the scenario.

6.3. Evaluation comments

Even the least effective heuristic studied (coarse-grain target) saves almost 40% of the intervisibility determinations for some scenarios; for example, Delay with Combat. It does this with high effectiveness. It can be argued that combat situations are where the heuristics are most needed because this is the most typical use of CGF systems.

The sighter and target-based heuristics seem very reasonable; for example, destroyed vehicles should not be sighted or attempt to sight. The other components of these composite heuristics are similarly reasonable, but time did not permit experimental validation of each component individually. The components taken together performed well.

Symmetry is shown to save on the order of 50% for all scenarios. This is a very intuitive result; because of the way symmetry works, one would expect it to eliminate half of the intervisibility determinations. The fact that this result was found in the experiment adds credibility to the experimental method used. Of course, the symmetry heuristic was tested only for entities that were generated by the same Simulator. Applying the symmetry heuristic across multiple Simulators, or in other words across multiple network nodes, would require network traffic to communicate the symmetry results. It is not at all clear whether the reduction in intervisibility processing produced by the symmetry heuristic is worth the additional network processing.

It may be wondered what effect, if any, would be seen if the implemented heuristics were combined to give more complex heuristics. Symmetry by itself is very successful. It may be that entities that have firepower kills do not need to be sighted. Combining symmetry and not doing intervisibility determinations to entities that have firepower kills can lead to a substantial savings in the number of intervisibility checks. There are many other such combinations that should be investigated. This project did not examine the heuristics in combination because of time constraints.

7. Conclusions and future work

7.1. Conclusions

For this project, a number of intervisibility heuristics were designed, implemented, and experimentally evaluated within a CGF system. The overall goal of the heuristics was to reduce the overall computational expense of intervisibility determination in CGF systems without materially affecting the realism of the autonomous behavior produced by those systems. Clearly, reducing computational load is a worthy goal. An overloaded system may perform poorly in demanding conditions resulting in behavior degradation and loss of realism.

The results show that the implemented intervisibility heuristics save substantial portions of the processing devoted to intervisibility checks, ranging from 10% to 50%.

Moreover, these savings were achieved at little cost in terms of CGF behavior realism. The behavior generated by the CGF system would be expected to suffer if an intervisibility heuristic significantly delayed the times at which hostile entities were sighted. That did not occur; the average sighting delay imposed by the various heuristics fell in the range of 0.3 to 0.5 seconds. Such a delay is negligible, especially in light of the tremendous savings in processing.

Computer generated forces are becoming increasingly complex as additional functionality is being added and more realistic behaviors are expected. It makes sense that some of the computational load be removed so that the CGF system can give more time to processing additional functions.

These results can be of great importance to CGF systems. By using one of these heuristics, the computational load of intervisibility determination can be greatly reduced, thereby freeing computational capacity that can be applied to generating more sophisticated behavior, performing more realistic physical modeling, or simply controlling more entities on a given system. Because these heuristics are completely independent of the terrain database format, they can be applied to any CGF system. Therefore, one or more intervisibility heuristics should be seriously considered for inclusion in any real-time CGF system.

7.2. Future work

Perhaps the biggest unanswered question, and, thereby, an opportunity for future work, is what might happen if the different heuristics were combined? Put another way, what would the effectiveness and cost be of the heuristics used in various combinations? Time constraints did not permit the examination of this issue in the project. Symmetry especially seems to be a likely candidate for combination with other heuristics , as its basic idea is very different from the other heuristics.

8. References

Barr, A. H. (1989). "Introduction to Physically-Based Modeling", Course #30, *ACM SIGGRAPH 89*, July 31 - August 4 1989, 5 pages.

Bresenham J. E. (1965), "Algorithm for Computer Control of Digital Plotter", *IBM Systems Journal*, Vol. 4, No. 1, pp. 25-30.

Cimini, F. C., Campbell, C. E., and Petty, M. D. (1992). "A Simple Flight Dynamics Model for Computer Generated Forces", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp 41-47.

Danisas, K., Smith, S. H., and Wood, D. D. (1990). "Sequencer/Executive for Modular Simulator Design", *Technical Report IST-TR-90-1*, Institute for Simulation and Training, University of Central Florida, 16 pages.

DIS Steering Committee (1993). "The DIS Vision: A Map to the Future of Distributed Simulation", *IST Technical Report*, 47 pages.

Gonzalez, G., Mullally, D. E., Smith, S. H., Vanzant-Hodge, A. F., Watkins, J. E., and Wood, D. D. (1990). "A Testbed for Automated Entity Generation in Distributed Interactive Simulation", *Technical Report IST-TR-90-15*, Institute for Simulation and Training, University of Central Florida, 37 pages.

Loper, M. L., Thompson, J. R., and Williams, H. L. (1991). "Simulator Networking: What Can It Offer The Training Community?", *Military Simulation & Training*, Issue 4 1991, pp. 11-14.

Petty, M. D., Campbell, C. E., Franceschini, R. W., Provost, M. H., and Karr, C. R. (1992a). "Preliminary Investigations into Efficient Line of Sight Determination in Polygonal Terrain", *Technical Report IST-TR-92-5*, Institute for Simulation and Training, February 28 1992.

Petty, M. D. (1992c). "Computer Generated Forces in Battlefield Simulation", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp. 56-71.

Pope, A. R. (1989). "The SIMNET Network and Protocols", *Report No. 7102*, BBN Systems and Technologies, July 1989, 160 pages.

Smith, S. H., Karr, C. R., Petty, M. D., Franceschini R. W., and Watkins, J. E. (1992a). "The IST Computer Generated Forces Testbed", *Technical Report IST-TR-92-7*, Institute for Simulation and Training, University of Central Florida.

Smith, S. H., and Petty, M. D. (1992b). "Controlling Autonomous Behavior in Real-Time Simulation", *Proceedings of the Southeastern Simulation Conference 1992*, The Society for Computer Simulation, Pensacola FL, October 22-23 1992, pp. 27-40.

Thorpe, J. A. (1987). "The New Technology of Large Scale Simulator Networking: Implications for Mastering the Art of Warfighting", *Proceedings of the 9th Interservice/Industry Training Systems Conference*, Orlando FL, November 30-December 2 1987, pp. 492-501.

A. Appendices

A.1. Acronyms and glossary

Table A.1-A Acronyms and glossary used in this report.

A ↦ B	The relation "A has a line of sight to B." (A ↦B implies B ↦ A).
Fine-grain	Intervisibility heuristics that use a relatively high update rate to approximate continuous intervisibility delays are termed "Fine-Grain" in this report.
FG	Fine-grain.
Intervisibility	Line Of Sight.
IUR	The Intervisibility Update Rate. This refers to the effective rate, as contrasted with the BUR.
His	The history heuristic.
Sym	The symmetry heuristic.
Sgt	Discrete version of the sighter-based heuristic.
Trg	Discrete version of the target heuristic.
Fgs	Fine-grained version of the sighter-based heuristic.
Fgt	Fine-grained version of the target-based heuristic.
Fgh	Fine-grained version of the history heuristic.
meeting	The meeting scenario.
assault	The assault scenario.
delay	The delay scenario.
reference file	The data file to be used by the Statistical Analyzer as the base or reference.
input file	The data file to be used by the Statistical Analyzer whose sighting data is to be compared to sightings in the reference file.
L0.5	No-heuristic run with BUD = 50 (BUR = 2, 2 checks/sec.)
L1.5	No-heuristic run with BUD = 150 (BUR = 0.67, 1 check every 1.5 seconds)
L2.0	No-heuristic run with BUD = 200 (BUR = 0.50, 1 check every 2.0 seconds)
L3.0	No-heuristic run with BUD = 300 (BUR = 0.33, 1 check every 3.0 seconds)
L4.0	No-heuristic run with BUD = 400 (BUR = 0.25, 1 check every 4.0 seconds)
L5.0	No-heuristic run with BUD = 500 (BUR = 0.20, 1 check every 5.0 seconds)
ARPA	Advanced Research Projects Agency
DARPA	Defense Advanced Research Projects Agency
DIS	Distributed Interactive Simulation
SIMNET	Simulation Network
CDRL	Contract Data Requirements List
IST	Institute for Simulation and Training
CGF	Computer Generated Forces
BMP	Boyevaya Mashina Pekhoty (Russian infantry fighting vehicle)

A.2. Statistical analyzer

To analyze the data, a tool was implemented to match reference data (results from running the Simulator without a heuristic) with test data (results from running a Simulator with a heuristic).

A.2.1. Data structures

The LOS (Line Of Sight) program analyzes files containing binary LOS records (generated by the modified Simulators). The format of each LOS record is specified in header file "LOS.H" and uses structures defined in the header file "PRIMITIV.H."

Each LOS record contains:

- Sighter ID number and host number,
- Sighter location x,y,z,
- Target ID number and host number,
- Target location x,y,z, and
- Time-of-sighting.

The time-of-sighting is an integer tick count that is based on the PC's normal clock rate of approximately 18.2 ticks per second.

The records in a LOS file should contain time values that are monotonically increasing.

The LOS program maintains internal data structures that store time as a 'C' integer with a maximum value 32,767 ticks. It maintains internal data structures that assume the ID numbers range from 0 through 11. An error is printed if an ID exceeds this limit. The internal data structures store x,y,z as 'C' unsigned integers with maximum values of 65,535 meters.

A.2.2. LOS program runtime options

The command to run the LOS program is:

los [-Wn[.m]] [-N] [-Lj[.k]] [-Dx] file_name1 [file_name2]

where:

file_name1	is a required binary file containing LOS_DATA records
file_name2	is an optional binary file containing LOS_DATA records
-Wn[.m]	sets an upper bound in seconds for the sighting window
-N	prevents test of an ID after it has caused a distance error
-Lj[.k]	sets an upper limit in seconds for times from the files
-Dx	sets an upper limit in meters for distances to allow a match

If only one LOS file is specified, the file is printed and single-file analysis is performed. The -W, -N, -L, and -D options are verified but are not used for this case.

If two LOS files are specified, double-file analysis is performed. The -W, -N, -L, and -D options are verified and used.

A.2.3. Single-File analysis

Each record in the LOS file is formatted and printed. The time in the LOS record is printed as seconds (tick count divided by 18.2, the ticks/second of the PC clock).

Sighters are identified simply by an ID number. The sighter's host number is not printed. Targets are identified by an ID number that is qualified by a host number. The "ID,host" combination is denoted B,h. The LOS program prints events as (A-->B,h).n where A and B are ID numbers, h is the target host number, and n is the count of A-->B,h events in the file containing A-->B,h from the beginning through the current event. During the print, statistics are maintained for each sighter-->target, (A-->B,h), pair. For each A-->B,h pair found, the values printed are:

- The time (in seconds) of the first A-->B,h sighting,
- The total number of A-->B,h sightings,
- The average time (in seconds) between successive A-->B,h sightings.

The total number of LOS records is also printed.

A.2.4. Double-File analysis

Double-file analysis requires two LOS data-file names to be specified when the LOS program is started. The first file named is used as the Reference-file, the second is the Test-file.

Analysis proceeds record-by-record through the Reference-file until either the end of the Reference-file is reached or a time greater than that specified by the 'L' option is reached. Each record indicates a sighting event and the time associated with the sighting.

Sighters are identified simply by an ID number. The sighter's host does not change within a file and is not printed. Targets are identified by an ID, qualified by a host number. The "ID,host" combination is denoted B,h. The LOS program prints events as (A-->B,h).n where A and B are ID numbers, h is the target host number, and n is the count of A-->B,h events in the file containing A-->B,h from the beginning through the current event. For each event not printed from the Reference-file:

- The default time-window is first determined. The default time-window begins at the time of the previous A-->B,h event in the Reference-file, or at time 0 if there is no previous A-->B,h event. The default time-window ends with the time of the next A-->B,h event in the Reference-file, or at arbitrary time of 32,000, if there is no next A-->B,h event.

- The default time-window may then be modified, if the -Wn[.m] parameter was specified. This parameter establishes an upper limit for the size of the time-window around the A-->B,h event as follows: if -Wn[.m] was specified, the beginning window time may be increased to the time of Event A-->B,h minus n.m. The ending window time may be decreased to the time of Event A-->B,h plus n.m. If specified, -Wn[.m] can only pull the ends of the time-window toward the A-->B,h time and so can only make the current default time-window smaller.

- All unprinted A-->B,h events in the Test-file that occurred before the desired time-window are printed. These Test-file A-->B,h events are identified as "extra", since no matching event is printed from the Reference-file. The "extra" Test-file events are marked internally as having been printed.

- The LOS program then attempts to find an A-->B,h event in the Test-file that is within the desired time-window surrounding the Reference-file A-->B,h event. If no A-->B,h event is found within the time-window, the Reference-file event is printed with an indication that the event was "missed" by the Test-file.

If, however, an A-->B,h event is found within the window, the distance between sighter A in the Reference-file and sighter A in the Test-file, and the distance between Target B,h in the Reference-file and Target B,h in the Test-file, are computed. These distances are used to determine if the A-->B,h pair found in the Test-file is the same A-->B,h pair from the Reference-file. One of two prints:

- If the sighter-to-sighter distance and the target-to-target distance are both smaller than the distance specified by -Dw, or if -Dw was not specified, the Reference-file (A-->B,h).n and the Test-file (A-->B,h).n events are printed, along with the event-time from each file and the difference in time between the sightings. This is considered to be a sighting compare.

- If the -Dw parameter was specified, and if either the sighter-to-sighter distance or the target-to-target distance is greater than the distance specified by -Dw, the Reference-file (A-->B,h).n and the Test-file (A-->B,h).n are printed along, with an error indicating that the events were "mismatched". The coordinates of the sighters and targets are printed. The Test-file event is marked internally as having been printed.

In either case, the Test-file event is marked internally as having being printed. If the -N parameter was specified, and if the sighter-to-sighter distance is greater than that specified by -Dw, all of the sighter A events in both the Reference and Test files are marked as printed. This eliminates sighter A from any further testing. If the -N parameter was specified, and if the target-to-target distance is greater than that specified by -Dw, all of the Target B,h events in both the reference and test files are marked as printed. This eliminates Target B from any further testing.

When all events from the Reference-file have been processed, the LOS program prints all events from the Test-file that were not previously printed. These are labeled as "extra" Test-file records.

Then, a count of the sightings compared (matched) is printed. For those sightings that compared, the mean and standard deviation of the signed time differences are printed. The mean and standard deviation of the absolute value of the time differences are also printed.

Finally, counts are printed:

- Reference-file sightings "missed" by the test-file,

- "Extra" sightings that were only in the Test-file, and

- Sightings where the events were mismatched (distance too great). This count is also expressed as a percentage of the sum of the compare count plus the mismatch count.

A.3. Skeleton for the fine-grain heuristics

The fine-grain baseline (or *skeleton*) exists as a template for developing fine-grain heuristics. It contains the watch-dog code, routines to allow statistics to be computed, and other elements common to all the fine-grain heuristics.

This baseline produces checks at the same rate the user requested by using the trivial "heuristic" which requests a skip to be done every time (this is overridden by the watch-dog at the user requested frequency).

The granularity is determined by a multiplier used in common by all the fine-grain heuristics. In brief, when an intervisibility message has been processed, a new intervisibility message is sent. The delay associated with this message is normally that specified by the user. For FG heuristics the delay is divided by the granularity multiplier, which is referred to as the *RATE*. A RATE of 3, for example, will not necessarily triple the message rate (which is the intent) for two reasons. Because more messages are being processed, delays may increase and, if RATE does not evenly divide the user requested delay, truncation will increase the rate. For example, if the user desires 20 messages a second, a value of 5 is given in the configuration file (it represents hundredths of seconds between deliveries). For the example at hand, the RATE used will not result in 60 messages a second but 100, because 5/3 will be treated as 1.

If the user requests N messages per second, and there are B *local* blue vehicles and R *local* red vehicles, the traffic produced will be N·B·R. Assuming no rounding, the RATE simply increases this to RATE·N·B·R. However, if the number of local vehicles is limited to 12 (as is the case for extant Simulator) B·R is bounded above at 36. The usual value of N is no greater than 1. Thus, the number of messages produced by intervisibility requests (assuming no round off and message delays can be ignored) is no more than 36·RATE.

It was decided only to consider RATEs which were powers of two: halving the interval between tests has a natural appeal. With that restriction, delays because of message flooding are not apparent, at least on the machines used for evaluation, until the RATE is 8 (36·8 = 288 messages per second, but in the usual case the round off done when dividing 100 by 8 yields a rate of 300 messages per second).

All the FG heuristics were based on RATE being 4 (4 messages per second when the user requests 1 update per second). Since the number of intervisibility messages generated has a factor of both RATE and the user requested rate, it should be apparent that if the user doubles the requested RATE message traffic may begin to impact the simulation.

It should be understood that to prevent actually increasing the number of intervisibility updates done, the number of consecutive skips is never allowed to drop below RATE - 1. Hence, a minimum of 3 skips were done for every intervisibility determination done.

A.4. Message delay in fine-grain heuristics

When the fine-grain heuristics were designed, it was recognized they would generate additional overhead outside of the heuristics code. For example, the Simulator executive would have to handle more traffic to deliver the extra messages involved in stepping intervisibility through finer time intervals. To account for this, the heuristic measurements gather information about all levels of overhead involved in supporting their implementation.

An unanticipated problem came about in connection with timer granularity. It took a considerable effort to fully understand the nature of this problem (as several red herrings were pursued), and no ideal solution was found to handling the problem.

The major manifestation of the problem is in the failure of the FG skeleton (Section A.3) to carry out as many raw heuristic updates as the no-heuristic model. Since the skeleton generates N times the traffic, but skips N-1 times, it should do the same number of updates regardless of the setting of N for $1 \leq N \leq M$ (where M is chosen such that the Simulator is not overwhelmed by the additional message traffic). This did not turn out to be correct.

Even with N = 2 under trivial system load, there was a significant drop in the number of intervisibility determinations completed. It had been decided at the outset that the skeleton versions of the heuristics should perform approximately the same number of intervisibility determinations as the no-heuristic system. To illustrate how severe the problem is, here are the results for a trivial scenario and various values of N (Table A.4-A):

Table A.4-A Rate vs. message delivery performance.

Heuristic	Messages Expected	Messages Received
None	360	359
Skeleton, RATE = 1	360	359
Skeleton, RATE = 2	720	650
Skeleton, RATE = 4	1440	1287
Skeleton, RATE = 8	2880	1656

Lower values of N decrease the problem, but make the experiments less "fine-grained." In the remaining discussion an N of 4 is assumed. The authors decided 4 was the minimum value for the fine-grain experiments to live up to their name.

All of these problems result from the clock granularity. The clock tics about 18.205 times per second, or about 0.054930 seconds between tics. On the other hand, the user is allowed to specify the timer interval in hundredths of seconds. Because of this, the user requests fall into equivalence classes. For example, a user-requested delay of 1, 2, 3, 4, or 5 hundredths will all be treated as requests of approximately 5.493 hundredths of a second. Equivalent delays lead to the same (within experimental error) number of messages being generated. The equivalence class notation was experimentally verified by running a simple scenario (2 vehicles, no movement or firing) for 30 seconds. For this scenario, a BUD setting of 100 (1 check per second) should yield exactly 60 messages, if their is no overhead for delivery or computing intervisibility.

Table A.4-B shows the user requested BUD settings, the equivalence class into which the time should fall, the number of messages the user expects to see delivered with the requested

rate, and the number of messages actually delivered. For the equivalence class, both the time (a multiple of 0.054930) and the expected message delivery count for that delay time are given. Note that, as one moves down the rows, the actual delivery approaches the equivalence class delivery. Because of overhead and rounding, the actual rates never reach the "expected" rate.

Table A.4-B Timer equivalence classes.

Requested Delay	Equivalence Class	Msgs User Expects	Actual # of Messages
0	0	Infinite	26497
1	5.4930/1092	6000	929
2	"	3000	931
3	"	2000	931
4	"	1500	932
5	"	1200	931
6	10.986/546	1000	481
7	"	857	----
8	"	750	----
9	"	667	----
10	"	600	473
11*	16.479	545	471
12	"	500	319
13	"	462	----
14	"	429	----
15	"	400	----
16	"	375	319
17	21.972/273	352	247
18	"	333	----
19	"	316	----
20	"	300	----
21	"	286	241
22*	27.465/218	273	241
23	"	261	193
24	"	250	----
25	"	240	193
26	"	231	----
27	"	222	193
28	32.958/182	214	169
29	"	207	----
30	"	200	169

* These boundary cells yield results in the next equivalence class because the time spent in dispatching and computing intervisibility makes the rates a multiple of (5.4930 + ε) rather than just 5.4930. Notice that all of the cases marked are on the nominal class edges.

A.5. Rate analysis for fine-grain heuristics

To generate the fine-grain heuristics accelerated message rates are required. However, a consequence of more traffic is increased overhead. This appendix examines the additional overhead and discusses the rational used in selecting a rate.

These tables were generated by running a reasonably complicated test scenario. In all cases the 486/66 PC used was reduced to a 486 compatible speed (approximately 25 MHz). This was necessary to gather reasonable statistics about the run, (since clocks sampling at higher speeds are too granular to catch the code in intervisibility checks). In this table:

RATE is the multiplier applied to the user's requested intervisibility rate. A value of 5 indicates the system will generate 5 intervisibility updates for every one expected by the user. However, when no heuristic is in place (but rather only the heuristic skeleton, see Section A.3) 4 of the 5 requests will be treated as "skips" (i.e., these will not generate a "real" intervisibility update).

MSGS is the number of intervisibility update messages generated during the test. If there were no overhead this figure would be proportional to RATE since the same scenario was used for all the tests.

RAW is the number of actual intervisibility determinations done. Since the FG heuristic skeleton was in use this figure would be constant but for system overhead.

TIME is the amount of time (expressed as a percentage) spent in intervisibility updates. This figure includes all levels of intervisibility processing.

EXEC indicates the percentage of intervisibility time spent at the executive's level. This time reflects dequeuing and delivering intervisibility update messages.

COMP indicates the percentage of intervisibility time spent in computing whether an intervisibility determination should be done. Because the skeleton heuristic is in use this time is as small as possible. Real FG Heuristics may spend significant time determining whether an intervisibility determination is called for.

RAW lists the time, as a percentage, spent doing actual intervisibility determinations. EXEC + COMP + RAW yields 100%.

USER indicates the number of intervisibility determinations requested by the user deduced by taking the number of computation calls made (this figure is not shown in the table) and dividing by the RATE. If there were no overhead to FG this figure would be constant.

Two sets of data are shown. The first represents data collection with a user requested rate of 4 updates per second (per vehicle). The other shows the data corresponding to a user request of 3 1/8 updates per second. Certain key arithmetic regarding message delay involves rounding in one case, but is exact in the other. The two experiments were done to help understand the effects of rounding in these computations.

The first data row (with RATE marked N/A) shows the result when no heuristic is used. No skipping, or code to support skipping, was in place. The changes from this line to the next

(RATE equal one) gives an indication of the overhead needed to put any FG heuristic in place, regardless of the heuristic's innate speed.

Table A.5-A LOS Requested (4 updates/second).

Rate	Msgs	Raw	Time	Exec	Comp	Raw	User
N/A	1048	6092	56.0	6.2	0.1	93.7	6092
1	1015	5892	54.6	5.4	0.6	94.1	5892
2	1492	4383	43.3	11.2	0.8	88.1	4363
4	2136	3162	36.5	19.1	2.0	79.0	3135
8	3026	2277	32.9	34.2	3.3	62.5	2277
16	3961	1506	28.0	49.0	6.4	44.6	1456

Table A.5-B LOS Requested (3 1/8 updates/second).

Rate	Msgs	Raw	Time	Exec	Comp	Raw	User
N/A	988	5733	53.1	5.4	0.6	94.0	5733
1	963	5583	52.5	5.7	1.0	93.3	5583
2	1412	4140	40.9	13.0	1.6	85.4	4122
4	1965	2924	35.5	17.9	1.8	80.4	2896
8	2438	1830	27.2	33.0	4.5	62.5	1800
16	3670	1386	25.9	44.6	3.7	51.7	1354

Based on these results rounding does not appear to be an important factor for the purpose of these experiments.

A.6. Ill-conditioned problem

When small perturbations in a system cause large changes in the system's performance the system is said to be *ill conditioned*. For the purposes of this paper, the system under consideration is the IST CGF Simulator in combination with the utilities used to analyze its performance. The small perturbations to this system, which demonstrated its ill conditioned nature, included small changes in message delivery times, due to clock granularity and network traffic delivery times.

Divergence can lead to vehicles following slightly different routes (because message delivery affects when a turn is executed), and the cumulative effects on the route changes ultimately has severe ramifications on sighting events.

A.7. Script files for the scenarios

The format of the script commands is somewhat cryptic. Briefly, a command consists of 4 parts. First, a delay, in seconds, before processing the command (this allows a gap between the execution of one command and the next). Second, a character or integer identifies the recipient of the command. Characters indicate simulation managers (for example, 'd' is the "display manager"). Integers identify a simulated entity and is the vehicle number (a component of its SIMNET vehicle ID). The first created will have a vehicle number of 0. Third, a single character gives the command. Fourth is a set of parameters to the command.

For example:

```
0 d 0 180 100
```

With no (0) delay, the display manager is to move the display to the coordinates (180,100).

```
1 i c m1 100 150
```

After a one second wait, the initialization manager is to create an M1 vehicle at the coordinates (100,150).

Lines in a script that begin with an asterisk (*) are simply printed on the screen. If a line contains leading white space, it is ignored. These lines are used for comments.

This section gives listings for the various test scenarios used for the system evaluation.

Table A.7-A shows the script files that were used to implement the various scenarios.

Table A.7-A Scenario breakup into script files.

Scenario	Movement-only		Combat	
	Blue	Red	Blue	Red
Meeting	m_blu.tst	m_red.tst	mc_blu.tst	c_red.tst
Delay	d_blu.tst	d_red.tst	dc_blu.tst	dc_red.tst
Assault	a_blu.tst	a_red.tst	ac_blu.tst	ac_red.tst

A.7.1. Script files for the meeting engagement scenario

These are non-combat scenarios. "RED" scripts handle the red forces which play the role of the master in the simulations. "BLU" scripts control the blue forces and play the role of the slave simulator.

M_BLU.TST

```
* S1 awaits m1
1 k w me 1

*
* Creation of first platoon: 4 X M2
* (right echelon after facing south)
*
0 d o 40080 42591
0 i c m2          * 0
0 d m 5 71
0 i c m2          * 1
0 d m 0 71
0 i c m2          * 2
0 d m -10 80
0 i c m2          * 3
*
* Creation of second platoon:
* 2 X M1, 1 X M109, 1 X M977
* (Wedge)
*
0 d o 40455 42631
0 i c m1          * 4
0 d m 72 75
0 i c m978        * 5
0 d m 78
0 i c m978        * 6
0 d m 71 -73
0 i c m1          * 7
*
* Creation of third platoon: 4 X M2
* (left echelon after facing south)
*
0 d o 41021 42550
0 i c m2          * 8
0 d m -70 70
0 i c m2          * 9
0 d m -70 70
0 i c m2          * 10
0 d m -70 70
0 i c m2          * 11
*
* Set facing
*
0 0 f 180
0 1 f 180
0 2 f 180
0 3 f 180
0 4 f 180
0 5 f 180
0 6 f 180
0 7 f 180
0 8 f 180
0 9 f 180
0 10 f 180
0 11 f 180
*
* Specify waypoints for the first
* platoon
*
1 0 pw 40135 41929 a
0 0 pw 40100 41446 a
0 0 pw 39984 40935 a
0 0 pw 40032 40419 a

1 1 pw 40135 41929 a
0 1 pw 40100 41446 a
0 1 pw 39984 40935 a
0 1 pw 40096 40491 a

1 2 pw 40135 41929 a
0 2 pw 40100 41446 a
0 2 pw 39984 40935 a
0 2 pw 40112 40571 a

1 3 pw 40135 41929 a
0 3 pw 40100 41446 a
0 3 pw 39984 40935 a
0 3 pw 40192 40605 a
*
* Specify waypoints for the second
* platoon
*
1 4 pw 40375 41612 a
0 4 pw 40566 41005 a
0 4 pw 40494 40500 a
0 4 pw 40495 40332 a

1 5 pw 40445 41682 a
0 5 pw 40636 41075 a
0 5 pw 40564 40570 a
```

```
0 5 pw 40550 40402 a

1 6 pw 40515 41682 a
0 6 pw 40706 41075 a
0 6 pw 40643 40570 a
0 6 pw 40645 40422 a

1 7 pw 40585 41612 a
0 7 pw 40776 41005 a
0 7 pw 40713 40500 a
0 7 pw 40745 40332 a
*
* Specify waypoints for the third
* platoon
*
1 8 pw 41464 42284 a
0 8 pw 41186 41242 a
0 8 pw 40902 40649 a

1 9 pw 41394 42354 a
0 9 pw 41116 41312 a
0 9 pw 40832 40719 a

1 10 pw 41324 42424 a
0 10 pw 41046 41382 a
0 10 pw 40762 40789 a

1 11 pw 41254 42494 a
0 11 pw 40976 41452 a
0 11 pw 40692 40859 a
*
* Slew display
*
1 d m -400 -1700
0 d s 7
*
* Tell platoons to move
*
1 0 gw
4 1 gw
3 2 gw
2 3 gw

2 4 gw
2 7 gw
4 5 gw
2 6 gw

2 8 gw
2 9 gw
2 10 gw
2 11 gw

1 6 a r
8 6 a n
*
*** end of script file
*
```

M_RED.TST

```
*
* move sim to desired location and
* set scale
*
0 d o 40212 39505
0 d s 8

30 ,
2 17 29 k e me 1
2 ,
*
* instantiation of red platoon 1
*
0 i c bmp2 39080 37540 0
0 i c bmp2 39010 37619 0
0 i c bmp2 38940 37680 0
0 i c bmp2 38870 37759 0
*
* instantiation of red platoon 2
*
0 i c t72 40212 37495 0
0 i c URAL375F 40282 37435 0
0 i c URAL375F 40352 37430 0
0 i c t72 40424 37495 0
*
* instantiation of red platoon 3
*
0 i c bmp2 41169 37589 0
0 i c bmp2 41099 37510 0
0 i c bmp2 41019 37449 0
0 i c bmp2 40949 37370 0
*
* waypoints for the key vehicle (0)
* in red platoon 1
*
0 0 pw 39440 38273
0 0 pw 39834 38951
0 0 pw 40023 39692
0 0 pw 40132 40100
*
* waypoints for the vehicle (1) in
* red platoon 1
```

```
*
0 1 pw 39374 38349
0 1 pw 39764 39021
0 1 pw 39979 39805
 0 1 pw 40062 40170
*
* waypoints for the vehicle (2) in
* red platoon 1
*
0 2 pw 39306 38413
0 2 pw 39694 39091
0 2 pw 39793 39663
0 2 pw 39883 39832
 0 2 pw 39992 40240
*
* waypoints for the vehicle (3) in
* red platoon 1
*
0 3 pw 39201 38387
0 3 pw 39234 38487
0 3 pw 39624 39161
0 3 pw 39813 39902
0 3 pw 39922 40310
*
* waypoints of KEY vehicle (4) in
* red platoon 2
*
0 4 pw 40164 38433
0 4 pw 40198 38700
0 4 pw 40231 39215
0 4 pw 40207 39362
0 4 pw 40123 39536
0 4 pw 40359 40056
*
* waypoints of vehicle (5) in red
* platoon 2
*
0 5 pw 40234 38503
0 5 pw 40301 39285
0 5 pw 40193 39606
0 5 pw 40354 39926
*
* waypoints of vehicle (6) in red
* platoon 2
*
0 6 pw 40304 38503
0 6 pw 40448 38627
0 6 pw 40371 39285
0 6 pw 40263 39676
0 6 pw 40446 39911
*
* waypoints of vehicle (7) in red
* platoon 2
*
0 7 pw 40346 38336
0 7 pw 40445 38634
0 7 pw 40441 39355
0 7 pw 40333 39746
0 7 pw 40515 39976
*
* waypoints of KEY vehicle (8) in
* red platoon 3
*
0 8 pw 41161 38317
0 8 pw 41191 38604
0 8 pw 40809 39213
0 8 pw 40617 39347
0 8 pw 41158 39685
0 8 pw 41210 40325
*
* waypoints of vehicle (9) in red
* platoon 3
*
0 9 pw 41110 38251
0 9 pw 41154 38545
0 9 pw 40739 39143
0 9 pw 40557 39270
0 9 pw 41088 39615
0 9 pw 41140 40255
*
* waypoints of vehicle (10) in red
* platoon 3
*
0 10 pw 41044 38170
0 10 pw 40779 38383
0 10 pw 40669 39073
0 10 pw 40487 39200
0 10 pw 40593 39406
0 10 pw 41018 39545
0 10 pw 41070 40185
*
* waypoints of vehicle (11) in red
* platoon 3
*
0 11 pw 40977 38075
0 11 pw 40735 38280
0 11 pw 40599 39003
0 11 pw 40417 39130
0 11 pw 40523 39336
0 11 pw 40948 39475
0 11 pw 41000 40115
*
* give red platoon 3 the command
* to move out
*
1 8 gw
1 9 gw
1 10 gw
```

```
1 11 gw
*
* give red platoon 1 the command
* to move out
*
10 2 gw
2 3 gw
1 1 gw
2 0 gw
*
* give red platoon 2 the command
* to move out
*
25 4 gw
1 5 gw
1 6 gw
1 7 gw

306 ,
5 17 29 k x
1 ,
1 exit
*
*** end of script file
```

A.7.2. Script files for the meeting engagement scenario with combat

MC_BLU.TST

```
* S1 awaits m1
1 k w mec 1

*
*  Creation of first platoon: 4 X M2
*  (right echelon after facing south)
*
0 d o 40080 42591
0 i c m2          * 0
0 d m 5 71
0 i c m2          * 1
0 d m 0 71
0 i c m2          * 2
0 d m -10 80
0 i c m2          * 3
*
*  Creation of second platoon: 2 X
*  M1, 1 X M109, 1 X M977
*  (Wedge)
*
0 d o 40455 42631
0 i c m1          * 4
0 d m 72 75
0 i c m978        * 5
0 d m 78
0 i c m978        * 6
0 d m 71 -73
0 i c m1          * 7
*
*  Creation of third platoon: 4 X M2
*  (left echelon after facing south)
*
0 d o 41021 42550
0 i c m2          * 8
0 d m -70 70
0 i c m2          * 9
0 d m -70 70
0 i c m2          * 10
0 d m -70 70
0 i c m2          * 11
*
*  Permission to fire
*
1 0 ( 4000
0 1 ( 4000
0 2 ( 4000
0 3 ( 4000
0 4 ( 4000
0 5 ( 4000
0 6 ( 4000
0 7 ( 4000
0 8 ( 4000
0 9 ( 4000
0 10 ( 4000
0 11 ( 4000
*
*  Set facing
*
0 0 f 180
0 1 f 180
0 2 f 180
0 3 f 180
0 4 f 180
0 5 f 180
0 6 f 180
0 7 f 180
0 8 f 180
0 9 f 180
0 10 f 180
0 11 f 180
*
*  Specify waypoints for the first
*  platoon
*
1 0 pw 40135 41929 a
0 0 pw 40100 41446 a
0 0 pw 39984 40935 a
0 0 pw 40032 40419 a

1 1 pw 40135 41929 a
0 1 pw 40100 41446 a
0 1 pw 39984 40935 a
0 1 pw 40096 40491 a

1 2 pw 40135 41929 a
0 2 pw 40100 41446 a
0 2 pw 39984 40935 a
0 2 pw 40112 40571 a

1 3 pw 40135 41929 a
0 3 pw 40100 41446 a
0 3 pw 39984 40935 a
0 3 pw 40192 40605 a
*
*  Specify waypoints for the second
*  platoon
*
1 4 pw 40375 41612 a
```

```
0 4 pw 40566 41005 a
0 4 pw 40494 40500 a
0 4 pw 40495 40332 a

1 5 pw 40445 41682 a
0 5 pw 40636 41075 a
0 5 pw 40564 40570 a
0 5 pw 40550 40402 a

1 6 pw 40515 41682 a
0 6 pw 40706 41075 a
0 6 pw 40643 40570 a
0 6 pw 40645 40422 a

1 7 pw 40585 41612 a
0 7 pw 40776 41005 a
0 7 pw 40713 40500 a
0 7 pw 40745 40332 a
*
* Specify waypoints for the third
* platoon
*
1 8 pw 41464 42284 a
0 8 pw 41186 41242 a
0 8 pw 40902 40649 a

1 9 pw 41394 42354 a
0 9 pw 41116 41312 a
0 9 pw 40832 40719 a

1 10 pw 41324 42424 a
0 10 pw 41046 41382 a
0 10 pw 40762 40789 a

1 11 pw 41254 42494 a
0 11 pw 40976 41452 a
0 11 pw 40692 40859 a
*
* Slew display
*
1 d m -400 -1700
0 d s 7
*
* Tell platoons to move
*
1 0 gw
4 1 gw
3 2 gw
2 3 gw

2 4 gw
2 7 gw
4 5 gw
2 6 gw

2 8 gw
2 9 gw
2 10 gw
2 11 gw

1 6 a r
8 6 a n
*
*** end of script
*
```

MC_RED.TST

```
*
* move display to desired location
* and set scale
*
0 d o 40212 39505
0 d s 8

30 ,
2 17 29 k e mec 1
2 ,
*
* instantiation of red platoon 1
*
0 i c bmp2 39080 37540 0
0 i c bmp2 39010 37619 0
0 i c bmp2 38940 37680 0
0 i c bmp2 38870 37759 0
*
* instantiation of red platoon 2
*
0 i c t72 40212 37495 0
0 i c URAL375F 40282 37435 0
0 i c URAL375F 40352 37430 0
0 i c t72 40424 37495 0
*
* instantiation of red platoon 3
*
0 i c bmp2 41169 37589 0
0 i c bmp2 41099 37510 0
0 i c bmp2 41019 37449 0
0 i c bmp2 40949 37370 0

0 0 ( 4000
0 1 ( 4000
```

```
0 2 ( 4000
0 3 ( 4000
0 4 ( 4000
0 5 ( 4000
0 6 ( 4000
0 7 ( 4000
0 8 ( 4000
0 9 ( 4000
0 10 ( 4000
0 11 ( 4000
*
* waypoints for the key vehicle (0)
* in red platoon 1
*
0 0 pw 39440 38273
0 0 pw 39834 38951
0 0 pw 40023 39692
0 0 pw 40132 40100
*
* waypoints for the vehicle (1) in
* red platoon 1
*
0 1 pw 39374 38349
0 1 pw 39764 39021
0 1 pw 39979 39805
0 1 pw 40062 40170
*
* waypoints for the vehicle (2) in
* red platoon 1
*
0 2 pw 39306 38413
0 2 pw 39694 39091
0 2 pw 39793 39663
0 2 pw 39883 39832
0 2 pw 39992 40240
*
* waypoints for the vehicle (3) in
* red platoon 1
*
0 3 pw 39201 38387
0 3 pw 39234 38487
0 3 pw 39624 39161
0 3 pw 39813 39902
0 3 pw 39922 40310
*
* waypoints of KEY vehicle (4) in
* red platoon 2
*
0 4 pw 40164 38433
0 4 pw 40198 38700
0 4 pw 40231 39215
0 4 pw 40207 39362
0 4 pw 40123 39536
0 4 pw 40359 40056
*
* waypoints of vehicle (5) in red
* platoon 2
*
0 5 pw 40234 38503
0 5 pw 40301 39285
0 5 pw 40193 39606
0 5 pw 40354 39926
*
* waypoints of vehicle (6) in red
* platoon 2
*
0 6 pw 40304 38503
0 6 pw 40448 38627
0 6 pw 40371 39285
0 6 pw 40263 39676
0 6 pw 40446 39911
*
* waypoints of vehicle (7) in red
* platoon 2
*
0 7 pw 40346 38336
0 7 pw 40445 38634
0 7 pw 40441 39355
0 7 pw 40333 39746
0 7 pw 40515 39976
*
* waypoints of KEY vehicle (8) in
* red platoon 3
*
0 8 pw 41161 38317
0 8 pw 41191 38604
0 8 pw 40809 39213
0 8 pw 40617 39347
0 8 pw 41158 39685
0 8 pw 41210 40325
*
* waypoints of vehicle (9) in red
* platoon 3
*
0 9 pw 41110 38251
0 9 pw 41154 38545
0 9 pw 40739 39143
0 9 pw 40557 39270
0 9 pw 41088 39615
0 9 pw 41140 40255
*
* waypoints of vehicle (10) in red
* platoon 3
*
0 10 pw 41044 38170
0 10 pw 40779 38383
0 10 pw 40669 39073
0 10 pw 40487 39200
```

```
0 10 pw 40593 39406
0 10 pw 41018 39545
0 10 pw 41070 40185
*
*  waypoints of vehicle (11) in red
*  platoon 3
*
0 11 pw 40977 38075
0 11 pw 40735 38280
0 11 pw 40599 39003
0 11 pw 40417 39130
0 11 pw 40523 39336
0 11 pw 40948 39475
0 11 pw 41000 40115
*
*  give red platoon 3 the command
*  to move out
*
1 8 gw
1 9 gw
1 10 gw
1 11 gw
*
*  give red platoon 1 the command
*  to move out
*
10 2 gw
2 3 gw
1 1 gw
2 0 gw
*
*  give red platoon 2 the command
*  to move out
*
25 4 gw
1 5 gw
1 6 gw
1 7 gw

306 ,
5 17 29 k x
1 ,
1 exit
*
*** end of script file
*
```

A.7.3. Script files for the delay scenario

D_BLU.TST

```
*
* Display
*
0 d s 8
0 d m 23365 22359

1 k w de 1
* S1 awaits m1

*
* Create platoon 1
*
0 i c m2 23383 23885
0 0 f 270
0 i c m2 23437 23936
0 1 f 270
0 i c m2 23483 23981
0 2 f 270
0 i c m2 23531 24137
0 3 f 270
*
* Create platoon 2
*
0 i c m1 23879 23141
0 4 f 270
0 i c m978 23949 23211
0 5 f 270
0 i c m978 24075 23117
0 6 f 270
0 i c m1 24014 22933
0 7 f 270

0 7 pg 22617 21315
6 4 pg 22715 21487
6 5 pg 22740 21401
6 6 pg 22699 21340

36 0 pg 22740 22296
6 1 pg 22740 22381
5 2 pg 22826 22259
3 3 pg 22936 22173

154 7 f 290
3 5 f 290
5 4 f 290
9 6 f 290
*
* Start the attack
*
9 4 a d
0 5 a d
0 6 a d
0 7 a d

0 0 f 270
0 1 f 270

0 4 pg 20852 22553
6 7 pg 20705 22259
4 5 pg 20889 22492
4 6 pg 20852 22369

5 2 f 270
*
* move out blue1
*
7 0 pg 21513 22663
2 1 pg 21575 23006
2 2 pg 21551 22859
3 3 pg 21514 22774
*
* Attack
*
0 0 a d
0 1 a d
0 2 a d
0 3 a d
```

D_RED.TST

```
*
* Set scale and init position
*
0 d o 22326 21805
0 d s 9

30 ,
2 17 29 k e de 1
2 ,
*
* create red platoon 1
*
0 i c t72 21025 24105
0 i c bmp2 20925 24105
0 i c ural375f 21000 24055
0 i c ural375f 20950 24055
0 i c bmp2 21025 24000
0 i c t72 20925 24000

0 0 f 90
0 1 f 90
0 2 f 90
0 3 f 90
0 4 f 90
0 5 f 90
*
* create BLUE platoon 3
*
0 i c m2 23547  21693
0 i c m2 23664  21765
0 i c m2 23735  21837
0 i c m2 23806  21904

0 6 f 270
0 7 f 270
0 8 f 270
0 9 f 270

0 6 ] 2
0 7 ] 2
0 8 ] 2
0 9 ] 2
*
* First RED Movement
*
0 5 pg 20674 23288
0 4 pg 20771 23288
3 3 pg 20705 23339
1 2 pg 20760 23334
3 0 pg 20774 23383
0 1 pg 20676 23389
*
* move BLUE platoon out
*
11 6 pg 22568 20495
2 7 pg 22630 20581
2 8 pg 22630 20519
2 9 pg 22679 20568
*
* Second RED Movement
*
45 0 pg 21667 23411
0 4 pg 21667 23306
3 2 pg 21617 23356
1 3 pg 21617 23331
3 1 pg 21567 23411
0 5 pg 21567 23306
*
* Third Red Movement
*
32 4 pg 21361 22389
0 5 pg 21261 22494
3 2 pg 21336 22444
0 3 pg 21286 22444
0 1 pg 21261 22389
2 0 pg 21361 22494

68 6 f 300
5 7 f 300

13 8 f 300
5 9 f 300

18 6 a d
0 7 a d
0 8 a d
0 9 a d
*
* Final BLUE phase
*
18 9 pg 21159 21573
2 6 pg 21086 21316
6 7 pg 21135 21500
2 8 pg 21110 21414
*
* Final RED phase
*
31 1 pg 20008 22723
4 2 pg 20058 22803
0 3 pg 20058 22753
4 4 pg 20108 22723
```

```
32 5 pg 20008 22828
0 0 pg 20108 22828

140 ,
```

```
5 17 29 k x
1 ,
1 exit
***end of script
```

A.7.4. Script files for the delay scenario with combat

DC_BLU.TST

```
*
* Display
*
0 d s 8
0 d m 23365 22359

1 k w dec 1
* S1 awaits m1

*
* Create platoon 1
*
0 i c m2 23383 23885
0 0 f 270
0 i c m2 23437 23936
0 1 f 270
0 i c m2 23483 23981
0 2 f 270
0 i c m2 23531 24137
0 3 f 270
*
* Create platoon 2
*
0 i c m1 23879 23141
0 4 f 270
0 i c m978 23949 23211
0 5 f 270
0 i c m978 24075 23117
0 6 f 270
0 i c m1 24014 22933
0 7 f 270
*
* Give Permission to Fire
*
1 0 ( 4000
0 1 ( 4000
0 2 ( 4000
0 3 ( 4000
0 4 ( 4000
0 5 ( 4000
0 6 ( 4000
0 7 ( 4000

0 7 pg 22617 21315

6 4 pg 22715 21487
6 5 pg 22740 21401
6 6 pg 22699 21340

36 0 pg 22740 22296
6 1 pg 22740 22381
5 2 pg 22826 22259
3 3 pg 22936 22173

154 7 f 290
3 5 f 290
5 4 f 290
9 6 f 290
*
* Attack
*
9 4 a d
0 5 a d
0 6 a d
0 7 a d

0 0 f 270
0 1 f 270

0 4 pg 20852 22553
6 7 pg 20705 22259
4 5 pg 20889 22492
4 6 pg 20852 22369

5 2 f 270
*
* move out blue 1
*
7 0 pg 21513 22663
2 1 pg 21575 23006
2 2 pg 21551 22859
3 3 pg 21514 22774
*
* Attack
*
0 0 a d
0 1 a d
0 2 a d
0 3 a d
```

DC_RED.TST

```
*
* set scale and init position
*
0 d o 22326 21805
```

```
0 d s 9

30 ,
2 17 29 k e dec 1
2 ,
*
* create red platoon 1
*
0 i c t72 21025 24105
0 i c bmp2 20925 24105
0 i c ural375f 21000 24055
0 i c ural375f 20950 24055
0 i c bmp2 21025 24000
0 i c t72 20925 24000

0 0 f 90
0 1 f 90
0 2 f 90
0 3 f 90
0 4 f 90
0 5 f 90
*
* create BLUE platoon 3
*
0 i c m2 23547  21693
0 i c m2 23664  21765
0 i c m2 23735  21837
0 i c m2 23806  21904

0 0 ( 4000
0 1 ( 4000
0 2 ( 4000
0 3 ( 4000
0 4 ( 4000
0 5 ( 4000
0 6 ( 4000
0 7 ( 4000
0 8 ( 4000
0 9 ( 4000

0 6 f 270
0 7 f 270
0 8 f 270
0 9 f 270

0 6 ] 2
0 7 ] 2
0 8 ] 2
0 9 ] 2
*
* First RED Movement
*
0 5 pg 20674 23288
0 4 pg 20771 23288
3 3 pg 20705 23339
1 2 pg 20760 23334
3 0 pg 20774 23383
0 1 pg 20676 23389
*
* move BLUE platoon out
*
11 6 pg 22568 20495
2 7 pg 22630 20581
2 8 pg 22630 20519
2 9 pg 22679 20568
*
* Second RED Movement
*
45 0 pg 21667 23411
0 4 pg 21667 23306
3 2 pg 21617 23356
1 3 pg 21617 23331
3 1 pg 21567 23411
0 5 pg 21567 23306
*
* Third RED Movement
*
32 4 pg 21361 22389
0 5 pg 21261 22494
3 2 pg 21336 22444
0 3 pg 21286 22444
0 1 pg 21261 22389
2 0 pg 21361 22494

68 6 f 300
5 7 f 300

13 8 f 300
5 9 f 300

18 6 a d
0 7 a d
0 8 a d
0 9 a d
*
* Final BLUE phase
*
18 9 pg 21159 21573
2 6 pg 21086 21316
6 7 pg 21135 21500
2 8 pg 21110 21414
*
* Final RED phase
*
31 1 pg 20008 22723
4 2 pg 20058 22803
0 3 pg 20058 22753
4 4 pg 20108 22723
```

```
32 5 pg 20008 22828
0 0 pg 20108 22828

140 ,
5 17 29 k x

1 ,
 exit
*
*** end of script file
*
```

A.7.5. Script files for the assault scenario

A_BLU.TST

```
*
* S1 wait
*
1 k w as 1

*
*  Creation of first platoon: 4 X M2
*  (right echelon after facing south)
*
0 d o 6870 6573
0 i c m2          * 0
0 d m 70 72
0 i c m2          * 1
0 d m 70 70
0 i c m2          * 2
0 d m 70 75
0 i c m2          * 3
*
*  Creation of second platoon:
*  2 X M1, 1 X M109, 1 X M977
*  (Wedge)
*
0 d o 7193 6853
0 i c m1          * 4
0 d m 70 -72
0 i c M978        * 5
0 d m 0 -69
0 i c m978        * 6
0 d m -70 -70
0 i c m1          * 7
*
*  Specify waypoints for the first
*  platoon
*
1 0 pw 6027 6911 a
0 0 pw 5404 6911 a
0 0 pw 4797 6826 a
0 0 pw 3726 6948 a
0 0 pw 3505 7782 a
0 0 pw 3179 9093 a

1 1 pw 6077 6988 a
0 1 pw 5454 6988 a
0 1 pw 4867 6896 a
0 1 pw 3790 7010 a
0 1 pw 3575 7852 a
0 1 pw 3249 9163 a

1 2 pw 6147 7021 a
0 2 pw 5504 7021 a
0 2 pw 4937 6966 a
0 2 pw 3850 7080 a
0 2 pw 3645 7922 a
0 2 pw 3319 9233 a

1 3 pw 6217 7064 a
0 3 pw 5554 7064 a
0 3 pw 5007 7026 a
0 3 pw 3930 7130 a
0 3 pw 3715 7992 a
0 3 pw 3389 9303 a
*
*  Specify waypoints for the second
*  platoon
*
1 4 pw 5875 7377 a
0 4 pw 5244 7872 a
0 4 pw 5140 8149 a
0 4 pw 4723 8799 a
0 4 pw 3988 8897 a
0 4 pw 3179 9093 a

1 5 pw 5945 7317 a
0 5 pw 5314 7802 a
0 5 pw 5210 8079 a
0 5 pw 4793 8729 a
0 5 pw 4058 8827 a
0 5 pw 3249 9023 a

1 6 pw 5945 7247 a
0 6 pw 5314 7732 a
0 6 pw 5210 8009 a
0 6 pw 4793 8659 a
0 6 pw 4058 8757 a
0 6 pw 3249 8953 a

1 7 pw 5875 7177 a
0 7 pw 5244 7662 a
0 7 pw 5140 7939 a
0 7 pw 4723 8589 a
0 7 pw 3988 8687 a
0 7 pw 3179 8883 a
*
*  Slew display
*
1 d o 6870 6593
0 d s 4
*
```

```
* Tell platoons to move
*
1 0 gw
1 1 gw
1 2 gw
1 3 gw

8 4 gw
1 7 gw
1 5 gw
1 6 gw

0 2 a r
8 2 a n
1 3 a r
5 3 a n
*
* Set speed
*
174 3 a d
8 2 a d
1 0 a d
0 1 a d
*
*** end of script file
*
```

A_RED.TST

```
*
* Red and blue platoon 3 Assault
* scenario
*
0 d s 9
0 d o 5238 8500

* start the simulation
30 ,
2 17 29 k e as 1
2 ,
*
* Create red force
*
0 i c bmp1 3765 9083
0 i c ural375f 3424 9162
0 i c bmp1 3647 8463
0 i c t72 3008 8921
0 i c ural375f 3407 9063
0 i c bmp1 3827 8334
*
* Create blue 3
*
0 i c m2 7120 7010
0 i c m2 7060 6922
0 i c m2 6988 6851
0 i c m2 6918 6791
*
* waypoints for blue 6 (key)
*
0 6 pw 6221 8311
0 6 pw 5238 8948
0 6 pw 4772 9301
0 6 pw 3814 9416
0 6 pw 3212 9680
*
* waypoints for blue 7
*
0 7 pw 6161 8251
0 7 pw 5168 8888
0 7 pw 4702 9231
0 7 pw 3764 9386
0 7 pw 3142 9620
*
* waypoints for blue 8
*
0 8 pw 6091 8181
0 8 pw 5098 8818
0 8 pw 4632 9161
0 8 pw 3714 9336
0 8 pw 3072 9550
*
* waypoints for blue 9
*
0 9 pw 6021 8111
0 9 pw 5028 8748
0 9 pw 4562 9091
0 9 pw 3664 9286
0 9 pw 3002 9480
*
* set speed
*
0 6 a n
0 7 a n
0 8 a n
0 9 a n
*
* Make red force face the blue
* force
*
1 0 f 90
0 1 f 100
0 2 f 90
0 3 f 90
```

```
0 4 f 90
0 5 f 120
*
*  have blue 3 move out
*
18 9 gw
3 6 gw
2 8 gw
2 7 gw
519 ,
5 17 29 k x
1 ,
1 exit
*
*** end of script file
*
```

A.7.6. Script files for the assault scenario with combat

AC_BLU.TST

```
*
* S1 wait
*
1 k w asc 1

*
* Creation of first platoon: 4 X M2
* (right echelon after facing south)
*
0 d o 6870 6573
0 i c m2          * 0
0 d m 70 72
0 i c m2          * 1
0 d m 70 70
0 i c m2          * 2
0 d m 70 75
0 i c m2          * 3
*
* Creation of second platoon: 2 X
* M1, 1 X M109, 1 X M977
* (Wedge)
*
0 d o 7193 6853
0 i c m1          * 4
0 d m 70 -72
0 i c M978        * 5
0 d m 0 -69
0 i c m978        * 6
0 d m -70 -70
0 i c m1          * 7
*
* Give Permission to fire
*
1 0 ( 4000
0 1 ( 4000
0 2 ( 4000
0 3 ( 4000
0 4 ( 4000
0 5 ( 4000
0 6 ( 4000
0 7 ( 4000
*
* Specify waypoints for the first
* platoon
*
1 0 pw 6027 6911 a
0 0 pw 5404 6911 a
0 0 pw 4797 6826 a
0 0 pw 3726 6948 a
0 0 pw 3505 7782 a
0 0 pw 3179 9093 a

1 1 pw 6077 6988 a
0 1 pw 5454 6988 a
0 1 pw 4867 6896 a
0 1 pw 3790 7010 a
0 1 pw 3575 7852 a
0 1 pw 3249 9163 a

1 2 pw 6147 7021 a
0 2 pw 5504 7021 a
0 2 pw 4937 6966 a
0 2 pw 3850 7080 a
0 2 pw 3645 7922 a
0 2 pw 3319 9233 a

1 3 pw 6217 7064 a
0 3 pw 5554 7064 a
0 3 pw 5007 7026 a
0 3 pw 3930 7130 a
0 3 pw 3715 7992 a
0 3 pw 3389 9303 a
*
* Specify waypoints for the second *
platoon
*
1 4 pw 5875 7377 a
0 4 pw 5244 7872 a
0 4 pw 5140 8149 a
0 4 pw 4723 8799 a
0 4 pw 3988 8897 a
0 4 pw 3179 9093 a

1 5 pw 5945 7317 a
0 5 pw 5314 7802 a
0 5 pw 5210 8079 a
0 5 pw 4793 8729 a
0 5 pw 4058 8827 a
0 5 pw 3249 9023 a

1 6 pw 5945 7247 a
0 6 pw 5314 7732 a
0 6 pw 5210 8009 a
0 6 pw 4793 8659 a
0 6 pw 4058 8757 a
0 6 pw 3249 8953 a

1 7 pw 5875 7177 a
```

```
0 7 pw 5244 7662 a
0 7 pw 5140 7939 a
0 7 pw 4723 8589 a
0 7 pw 3988 8687 a
0 7 pw 3179 8883 a
*
* Slew display
*
1 d o 6870 6593
0 d s 4
*
* Tell platoons to move
*
1 0 gw
1 1 gw
1 2 gw
1 3 gw

8 4 gw

1 7 gw
1 5 gw
1 6 gw

0 2 a r
8 2 a n
1 3 a r
5 3 a n
*
* Set speed
*
174 3 a d
8 2 a d
1 0 a d
0 1 a d
*
*** end of script file
*
```

AC_RED.TST

```
*
* Red and blue platoon 3 Assault
* scenario
*
0 d s 9
0 d o 5238 8500
*
* Start the simulation
*
30 ,
2 17 29 k e asc 1
2 ,
*
* Create red force
*
0 i c bmp1 3765 9083
0 i c ural375f 3424 9162
0 i c bmp1 3647 8463
0 i c t72 3008 8921
0 i c ural375f 3407 9063
0 i c bmp1 3827 8334
*
* create blue 3
*
0 i c m2 7120 7010
0 i c m2 7060 6922
0 i c m2 6988 6851
0 i c m2 6918 6791

0 0 ( 4000
0 1 ( 4000
0 2 ( 4000
0 3 ( 4000
0 4 ( 4000
0 5 ( 4000
0 6 ( 4000
0 7 ( 4000
0 8 ( 4000
0 9 ( 4000
*
* waypoints for blue 6 (key)
*
0 6 pw 6221 8311
0 6 pw 5238 8948
0 6 pw 4772 9301
0 6 pw 3814 9416
0 6 pw 3212 9680
*
* waypoints for blue 7
*
0 7 pw 6161 8251
0 7 pw 5168 8888
0 7 pw 4702 9231
0 7 pw 3764 9386
0 7 pw 3142 9620
*
* waypoints for blue 8
*
0 8 pw 6091 8181
0 8 pw 5098 8818
0 8 pw 4632 9161
0 8 pw 3714 9336
```

```
0 8 pw 3072 9550
*
*  waypoints for blue 9
*
0 9 pw 6021 8111
0 9 pw 5028 8748
0 9 pw 4562 9091
0 9 pw 3664 9286
0 9 pw 3002 9480
*
*  set speed
*
0 6 a n
0 7 a n
0 8 a n
0 9 a n
*
*  Make red force face the blue
*  force
*
1 0 f 90
0 1 f 100
0 2 f 90
0 3 f 90
0 4 f 90
0 5 f 120
*
*  have blue 3 move out
*
18 9 gw
3 6 gw
2 8 gw
2 7 gw

519 ,
5 17 29 k x
1 ,
1 exit
*
*** end of script file
*
```

A.8. The LOS journal

The LOS journal was written by Sumeet Rajput and portrays the ideas and thoughts that went behind this project. At the end of every day, the author would try to remember the important events of the day and record his feelings about how the project was progressing.

The journal reflects how problems were recognized and tackled. By its nature the journal is informal but nonetheless will help the reader understand some of the technical challenges that were overcome.

Interested readers can contact Mikel D. Petty to request access to the journal.