

A Mathematical Framework for Asynchronous, Decentralized, Decision-Making Algorithm with Semi-Autonomous Entities: Synthesis, Simulation, and Evaluation

Tony Lee, Sumit Ghosh
Department of Computer Science & Engineering
Arizona State University, Tempe, AZ 85287
tsl@enpc732.eas.asu.edu
sumit.ghosh@asu.edu

Jin Lu, Xiaolin Ge, Anil Nerode Wolf Kohn
Mathematical Sciences Institute Intermetrics, Inc.
Cornell University, Ithaca, NY 14850 Bellevue, WA 98004

Abstract

For many military and civilian large-scale, real-world problems of interest, data is first acquired asynchronously, i.e., at irregular intervals of time, at geographically-dispersed sites, processed utilizing decision-making algorithms, and the processed data then disseminated to other appropriate sites. Examples include the battlefield where intelligence data is gathered from different sources and processed to yield decisions and the proposed intelligent vehicle highway system where incident-related data is acquired along the highways to determine the best routes for traffic. The traditional approach to such problems consists of designing a central entity which collects all data and executes a decision making algorithm sequentially to yield the decisions. Centralized decision making algorithms are slow and highly vulnerable to natural and artificial catastrophes. Recent literature contains proposals for asynchronous, distributed, decision making algorithms wherein local decision making at every site replaces the centralized decision making to achieve faster response, higher reliability, and greater accuracy of the decisions. Two key issues with asynchronous, distributed, decision making algorithms include the lack of a suitable mechanism to synthesize them for any given problem and a comparative analysis of the quality of their decisions.

This paper proposes a mathematical framework, MFAD, based on the Kohn-Nerode distributed hybrid control paradigm, to describe a centralized decision-making algorithm and to synthesize from it a distributed decision-making algorithm. Ideally, the centralized control gathers all

data from all entities in the system and utilizes them to arrive at the decisions and, as a result, the decisions are expected to be “globally” optimal. In truth, however, as the frequency of the sensor data increases and the environment gets larger, dynamic, and more complex, the decisions, come into question. In the distributed decision-making system, the centralized decision-making is replaced by those of the constituent entities that aim at minimizing a Lagrangian, i.e., a local, non-negative cost criterion, subject to the constraints imposed by the global goal. Thus, computations are carried out locally, utilizing locally obtained data and appropriate information that is propagated from other sites. It is hypothesized that with each entity engaged in optimizing its individual behavior, asynchronously, concurrently, and independent of other entities, the distributed system will achieve “global” optimal behavior. The distributed system is also expected to be more efficient, robust, and exhibit scalability. While it does not claim that decentralized systems may be synthesized for all centralized real-world systems, this paper implements both the centralized and distributed paradigms for a representative military battlefield command, control, and communication (C^3) problem. It also simulates them on a testbed of network of workstations for a comparative performance evaluation of the centralized and decentralized paradigms in the MFAD framework. Performance results indicate that the decentralized approach consistently outperforms the centralized approach. The paper aims at developing a quantitative evaluation of the quality of decisions under the decentralized paradigm and, to achieve this goal, it introduces a fundamental concept, embodied through a hypothetical entity termed “Perfect Global Optimization Device (PGOD),” that generates perfect or ideal decisions. PGOD possesses perfect knowledge, i.e., the state information of every entity of the entire system unaffected by time and delay, at all times. PGOD utilizes the same decision-making algorithm as the centralized paradigm and generates perfect globally-optimal decisions which, though unattainable, provide a fundamental and absolute basis for comparing the quality of decisions. Simulation results reveal that the quality of decisions in the decentralized paradigm are superior to those of the centralized approach and that they closely track PGOD’s decisions.

Keywords: Optimization, distributed optimization, military command and control, distributed algorithms, synthesis of decentralized algorithms, asynchronous distributed algorithms, decision-making, perfect knowledge, perfect global optimization

1 Introduction

This paper is interested in an emerging class of real-world problems – asynchronous, distributed, decision-making algorithms, that aim to replace the traditional centralized decision-making approaches for efficiency, robustness, and scalability. In such problems, that span a wide geographical area, data is generated at different sites, processed locally utilizing local data and appropriate information obtained from other sites, and then disseminated to other sites. While each site is responsible for its own decision-making, independently and

asynchronously, the sum total of all decisions are expected to be consistent and meet the global goals. Examples of such problems include DARYN [1], DICA system [2], AMPRed [3], and HDDI [4]. The limitations of the traditional centralized algorithms are that they are extremely vulnerable to failure and are non-scalable. That is, as the number of entities in the system increases, the ability of the central agent to make timely decisions decreases. This is due to the increased computational load placed on the central agent and to the increased communications overhead associated with increased number of entities. While the asynchronous, distributed algorithms successfully address the above issues, key limitations include the lack of a formal approach to synthesize such algorithms, the need to coordinate the decisions of the entities to achieve global optimal behavior for the overall system, and the absence of a basis to rate the quality of their decisions. The literature on synthesizing decentralized decision-making algorithms and evaluating the quality of distributed decision-making, is sparse. Rotithor [7] proposes to distribute the overall tasks – system state estimation and decision making, of a decentralized decision-making system among independent decision-making processing elements and attempt to improve the overall performance of the system by optimizing the performance of the individual decision makers. Rotithor models the problem of load balancing in distributed computing and notes substantial performance improvements. Tsitsikilis and Athans [8] considers the distributed team decision problem wherein different agents obtain different stochastic measures related to an uncertain random vector from the environment and attempt to converge, asymptotically, on a common decision, and they derive the conditions for convergence. In [9], Tsitsikilis and Athans study the complexity of basic decentralized decision problems that are variations of “team decision problems” and conclude that optimality may be an elusive goal. It is noted that in contrast to the problems studied in [8, 9], in this paper, each entity derives its own decisions – possibly different from those of other entities – while attempting to achieve “global” optimal behavior.

This paper is based on the distributed hybrid control paradigm of Kohn and Nerode [5, 6] in that every entity of the decentralized decision-making system is designed to optimize a Lagrangian [11], a local, dynamically evolving, non-negative cost criterion, subject to the constraints imposed by the global objectives and principles. Every entity utilizes its local information, encapsulated through an internal state vector, its local goals, and appropriate coordination information obtained from other entities, to determine its decisions autonomously, independently, asynchronously, yet cooperatively, so as to achieve the desirable global performance. Compared to the centralized control system, the distributed decision-making system can process information faster and react to dynamic changes to the environment quicker. However, since each entity computes decisions autonomously, key global information must be made available to them. This is precisely the objective of the distributed Lagrangians. The distributed system is expected to exhibit robustness, i.e., resistance to catastrophic failure and scalability. That is, as the system grows in size due to increasing number of entities, the number of computing engines will increase proportionately and

the system is expected to continue to deliver undiminished performance. The joint state of all entities is the operative evolving state of the entire system and is referred to as the “carrier manifold” state in [10].

The paper is organized as follows. Section 2 describes a representative military command and control problem that will be modeled in the proposed mathematical framework. While section 3 formulates it in the centralized paradigm, section 4 presents a decomposition into the distributed paradigm. Section 5 describes the simulation architecture for the representative battlefield C^3 scenario employed in this paper. Section 6 presents an analysis of the performance results and section 7 presents some general conclusions.

2 A Representative Military Command, Control, and Communications Problem

The scenario presented here is a simulated battlefield with two opposing forces. Both forces consist of a number of identical “tank” units. However, while one uses centralized command and control structure, the other utilizes a decentralized structure. Under centralized control, each tank gathers information on the environment – detection and classification of enemy objects, tracking, etc., stores them in its internal state vector, and propagates the vector to the centralized decision-maker – a tank commander. Upon receiving the state vectors from all its constituent tanks, the commander processes the information, assesses the threats, computes the decisions, and propagates them to the individual tank units which, in turn, fires on the enemy. Under decentralized command and control, every tank utilizes the locally obtained data, coordination information from its peers, assesses threats to itself, and computes its own decisions, subject to the global goals that are manifested through its local goals. An entity is also capable of propagating information including threats, to a peer especially when the enemy is outside the field of view of the peer.

2.1 Component Vectors of the Entities

Each entity necessarily maintains a vector that consists of three component vectors to capture movement, sensor, and targeting information.

2.1.1 Movement state vector

The movement state vector contains information pertaining to the movement of this entity. It includes the current position of the unit, (x, y) , its velocity, (v_x, v_y) , and the amount of fuel remaining. Also included is the *status* of the unit which can be either *OPERATIONAL* or *DESTROYED*. The *mode* field represents the unit’s mode of operation. While the mode is normally *ENGAGE*, it may also be *RETREAT* or *SEARCHING*. When an entity is under enemy fire and is unable to “see” the enemy unit, it enters the *SEARCHING* mode which

Figure 1: Sensor Arc and Sensor Axis for an Entity

triggers its sensor management decisions in an attempt to locate the enemy unit in its field of view.

$$V_m = [status, fuel, mode, x, y, v_x, v_y]^T,$$

where T represents the transpose operation.

2.1.2 Sensor state vector

Each entity possesses sensor capability to view a limited area of its surroundings. This ability is defined through a cone originating at the unit and extending out for a finite distance. The cone is defined through two angles that form it, α and β , in a given coordinate system. The sensor axis for the cone is derived from the α and β , as shown in Figure 1.

Thus, the sensor state vector consists of the (α, β) pair which describes the orientation of the sensor arc.

$$V_s = [\alpha, \beta]^T,$$

where T represents the transpose operation.

2.1.3 Target state vector

The target state vector contains information on the state of the unit's weapon system and the position of its current target. While the *weaponstatus* can be either *READY* or *RELOADING*, the *ammunition* field reflects the number of shells remaining. The symbol τ represents the time at which the unit will open

fire on its target. The purpose of utilizing τ is elaborated later in the paper. The target itself is identified by its current coordinate position, (t_x, t_y) .

$$V_t = [weaponstatus, ammunition, t_x, t_y, \tau]^T,$$

where T represents the transpose operation.

2.2 State Vectors of the Entities

The state vector for an entity encapsulates its total knowledge of the system. Its components are presented subsequently. It is noted that while only the leader unit in the centralized paradigm must maintain the status of all friendly and sighted enemy units, every unit in the decentralized paradigm is a decision maker and must contain similar information.

2.2.1 Self state vector

The self state vector contains the state of this unit. Since the unit always has immediate access to its own status, this information is accurate up to the current time.

$$X_s = [V_m, V_s, V_t]$$

2.2.2 Friend state vector

Given n friendly units, the friend state vector must contain the state of each one of them. Since this information is obtained from the respective friendly units, it is necessarily delayed by a communication delay, τ_c . Thus, at the current time, T , the friend state vector represents the states of the friend units at time, $T - \tau_c$.

$$X_f = [V_{m,1}, V_{s,1}, V_{t,1}, \dots, V_{m,n-1}, V_{s,n-1}, V_{t,n-1}]$$

2.2.3 Enemy state vector

Where p enemy units are sighted, the enemy state vector contains the states of p enemy units. This information constitutes a fusion of what this unit can “see” through its own sensor and the enemy sighting reports that it has received from other friend units. Consequently, while a part of the information represents the enemy state at the current time T , the remainder of the vector represents the enemy state at time, $T - \tau_c$.

$$X_e = [V_{m,1}, V_{s,1}, V_{t,1}, \dots, V_{m,p}, V_{s,p}, V_{t,p}]$$

2.3 Complete Internal State Vector of an Entity

Thus, the complete internal state vector of every unit in the decentralized paradigm and the leader unit in the centralized approach, representing the unit’s entire knowledge of the world, may be expressed as a collection of the self, friend, and enemy state vectors.

$$X = [X_s, X_f, X_e]$$

2.4 Decision Vectors of Entities

A unit's decision vector captures the decisions computed by each unit in the decentralized case or the orders enforced on a unit in the centralized scheme by its leader. The components relate to movement decision, sensor management, and target management.

2.4.1 Movement Decision

A unit's physical movement is specified through two velocity vectors, v_x and v_y . The time rate of change of these vectors represents the acceleration of the unit and it is governed by its maximum acceleration or deceleration capability.

$$U_m = [v_x, v_y]^T$$

where T represents the transpose operation.

2.4.2 Sensor Management

As described earlier, the sensor arc is represented through α and β and is characterized by the sensor axis. Sensor management consists in changing the sensor axis, which is defined in degrees per second, subject to a maximum rate. The sensor decision vector is expressed as:

$$U_s = [\alpha, \beta]^T,$$

where T represents the transposition operation.

2.4.3 Target Management

Each tank unit has a turret mounted gun which is used to engage targets. This paper also assumes that a unit can open fire at an enemy unit only when it is within its field of view. Following target selection, it may be necessary to choose an optimal time, τ , to open fire. If the distance between the unit and the target is decreasing, then delaying the firing would increase the probability of hitting the target. However, this delay also increases the threat that the enemy unit presents to this unit and possibly other friendly units. In contrast, when the distance between the unit and the target is decreasing, it may be necessary to open fire immediately. Assuming target location at (t_x, t_y) , the target decision vector is written as:

$$U_t = [t_x, t_y, \tau]^T,$$

where T represents the transpose operation.

3 Modeling the C^3 Problem Under the Centralized Paradigm in MFAD

In the centralized paradigm, each individual unit communicates exclusively with the leader unit and there is a total absence of communication between the indi-

vidual tanks. Information relative to the environment flows from the individual units to the leader which computes the decisions for all other units and propagates them in the form of orders. Thus, the individual tanks need not maintain either friend or enemy state vectors and are not permitted to modify their own decision vectors. The leader unit must determine its own decision vector, U_m^1, U_s^1, U_t^1 , where the superscript 1 relates to the leader unit, and the decision vectors, U_m^i, U_s^i , and U_t^i for each of its $i = 2$ to n subordinate units. The complete decision vector is expressed as:

$$U = [U_m^1, U_s^1, U_t^1, \dots, U_m^n, U_s^n, U_t^n]$$

3.1 Synthesizing Decision Vectors through Cost Functions

To enable comparative analysis of the two paradigms, the respective decision vectors must be translated into real values. To achieve this goal, Lagrangians or cost functions, i.e., non-negative real valued functions, are developed corresponding to each of the component decision vectors of the entities – movement, sensor, and targeting. Unless otherwise noted, the component decisions are independent of each other. Therefore, optimal choices for each component of U may be obtained through minimizing the corresponding cost function.

3.1.1 Cost Function Corresponding to the Movement Decision

The cost function corresponding to the movement decision, J_m , is synthesized from (i) the distance D^i between the i th friendly unit and all sighted enemy units, for all friendly units, (ii) the threat rating T^i of all sighted enemy units against the i friendly unit, for all friendly units, and (iii) the deviation V^i of every i th friendly unit from the “movement axis” which is defined as the preferred direction of movement for all friendly units, as defined prior to the engagement. Thus,

$$J_m(X, U_m) = \sum_{i=1}^n J_m^i(X, U_m^i), \text{ where} \quad (1)$$

$$J_m^i(X, U_m^i) = \alpha_m^i \cdot D^i + \beta_m^i \cdot T^i(X) + \gamma_m^i \cdot V^i, \quad (2)$$

$$D^i = \sum_{j=1}^p \text{distance from the } i\text{th friendly unit to the } j\text{th enemy unit,}$$

$$T^i = \sum_{j=1}^p \text{distance from the } i\text{th friendly unit to the } j\text{th enemy unit,}$$

$$V^i = \text{actual direction of the } i\text{th friendly unit – movement axis,}$$

and α_m^i, β_m^i , and γ_m^i are constant weights that are assigned initial values at the beginning of the simulation and may be altered to influence the behavior of the i th friendly unit. In this research, all units, friend and enemy, are assumed to be identical and, therefore, the threat function is proportional to the distance

between the friend and enemy units. Should the need arise to model enemy units different from the friend units, the threat presented to the i th friendly unit by an enemy unit may be modified in the above cost function. It is also pointed out that there are p enemy units.

3.1.2 Cost Function Corresponding to the Target Decision

The Lagrangian for the target decision, J_t , is synthesized from two factors:

- The threat rating T^i posed by all sighted enemy units to the i th friendly unit,
- The probability P^i of the i th friendly unit securing a direct hit on its select target and destroying it. Thus

$$J_t(X, U_t) = \sum_{i=1}^n J_t^i(X, U_t^i), \text{ where} \quad (3)$$

$$J_t^i(X, U_t^i) = \alpha_t^i \cdot T^i + \beta_t^i \cdot P^i. \quad (4)$$

T^i , the threat rating, is defined earlier, and α_t^i and β_t^i are constant weights that are assigned initial values at the beginning of the simulation and may be altered to influence the behavior of the i th friendly unit.

3.1.3 Cost Function Corresponding to the Sensor Decision

The sensor decision cost function, J_s , is derived from four factors:

- The degree of overlap between the friendly sensors.
- The deviation of the sensor axis from a predefined threat axis, which is the expected direction of arrival of the enemy.
- A friendly unit's preference to include the nearest sighted enemy unit, not currently targeted by any friendly unit, within its sensor arc.
- An i th friendly unit's preference to include the nearest sighted enemy unit, currently targeted by another friendly unit, within its sensor arc. When the i th unit is also the current targeting unit, the preference is stronger and this scenario is one where the sensor decision is linked to a previous targeting decision.
- In addition, there is a special case when the unit has been fired upon by an unseen enemy, as described earlier, and it enters the *SEARCHING* mode. Given that all units are assumed to possess identical range, this guarantees the presence of an enemy unit within the range of the unit's weapon system. The *SEARCHING* mode encourages the unit to undergo

maximum change in its sensor arc to “see” new areas and locate the enemy. Thus,

$$J_s(X, U_s) = \sum_{i=1}^n J_s^i(X, U_s^i), \text{ where} \quad (5)$$

$$J_s^i(X, U_s^i) = \alpha_s^i \cdot O + \beta_s^i \cdot S + \gamma_s^i \cdot W + \delta_s^i \cdot V + \epsilon_s^i \cdot D. \quad (6)$$

O is an estimate of the degree of sensor overlap, S is the deviation of the sensor axis from the threat axis, W is the angle between the nearest, sighted, engaged enemy and the sensor axis, V is the angle between the nearest sighted unengaged enemy and the sensor axis, and α_s^i , β_s^i , γ_s^i , δ_s^i , and ϵ_s^i are constant weights that are assigned initial values at the beginning of the simulation and may be altered to influence the behavior of the i th friendly unit. In *SEARCHING* mode, D is assigned a very high value. At all other times, D is assigned the value 0.

3.1.4 Centralized Lagrangian

Thus, the overall cost function under the centralized paradigm, referred to as the centralized Lagrangian, is expressed subsequently, and the combinatorial optimization problem is to minimize it.

$$J(X, U) = J_m(X, U_m) + J_s(X, U_s) + J_t(X, U_t). \quad (7)$$

Upon inspection, the overall minimization effort, $\min\{J(X, U)\}$, may be rewritten in terms of simultaneously minimizing the n cost functions corresponding to the n individual friendly entities, as shown in equation (8). The result of minimization is the global minimum,

$$\min J(X, U) = \min\left\{\sum_{i=1}^n J^i(X, U^i)\right\} \quad (8)$$

The expression for minimizing the cost function of each entity may be elaborated to $\min J^i(X_s(t), X_f(t), X_e(t), U^i(t))$, with all of its components and the time dependency shown. The minimization effort for every entity is carried out by the leader unit, under the centralized paradigm. At a given time $t = T$, given the geographical separation between the leader and other entities, the leader unit is only aware of a friendly entity’s state up to $t = T - \tau_c$, where τ_c represents the communication delay between the leader and the entity in question. Thus, the peer state information suffers a latency of τ_c . While the state information for the entity itself is current, the enemy state information is a combination of information from its own sensors at $t = T$ and information sent from its peers at $t = T - \tau_c$. Therefore, the minimization effort corresponding to every entity is given by

$$\min J^i(X_s(T), X_f(T - \tau), X_e(T, T - \tau), U^i(T)) \quad (9)$$

It is pointed out that, following minimization, the decision vectors U^i that are generated must be propagated as orders to the subordinate units which execute them at $t = T + \tau_c$.

Figure 2: Interconnection under the Centralized and Decentralized Paradigms

4 Synthesizing Distributed Lagrangians for the C^3 Problem in MFAD

Under the distributed paradigm, every entity is granted autonomy to determine its own decision vector, locally. For the decisions to be effective, each entity must maintain state vectors for each of the friend and enemy units. This, in turn, requires every unit to communicate directly with its peers in addition to communicating with the leader unit. Thus, the status and sighting reports are exchanged directly between peers and with the leader unit. As noted earlier, information from a peer will suffer latency due to finite communication delay. Also, the leader unit is no longer responsible for executing the decision vectors of the subordinate units.

The Lagrangian corresponding to the i entity is synthesized analogous to the expression (9), except for three significant differences that are elaborated subsequently. The differences arise as a result of the decentralization of autonomy. The Lagrangian for the i th entity is presented in expression (10), where X_s^i , X_f^i , and X_e^i represent its internal state, its knowledge of the states of its peers, and its knowledge of the states of the enemy units.

$$J^i(X_s^i(T), X_f^i(T - \tau), X_e^i(T, T - \tau), U^i(T)) \quad (10)$$

First, in the centralized paradigm, the leader unit must determine the decision vectors for each of the n subordinates. Despite significant computational cost, the decision vectors are consistent, subject to the latency of the information from the subordinate units, since the leader has access to the state of the total system. In contrast, in the decentralized paradigm, every entity must compute only its own decision vector utilizing accurate knowledge of its state at the current time, knowledge of its peers' states up to time $t = T - \tau_e$, and the knowledge of the states of the enemy units acquired through its own sensors at the current time as well as from its peers at time $t = T - \tau_e$. None of the entities have access to the entire system state. Thus, while the computation

cost is lower, the units' decision vectors, computed at $t = T$ are not guaranteed to be coordinated.

Second, while there is execution latency in the centralized paradigm stemming from the need to propagate the decision vectors from the leader unit to the subordinates, the decision vector is executed immediately by the corresponding unit in the decentralized paradigm.

Third, the inter-peer connectivity and the messages exchanged between them, subject to the controlling parameter `INFORM_RANGE`, reflect a significant increase in communication of the decentralized paradigm over the centralized paradigm. Only when an enemy unit is sighted within `INFORM_RANGE` of the last known position of a friendly unit i , the sighting friendly unit j will propagate its sighting information to unit i .

5 A Representative Battlefield Scenario and the Simulation Architecture

A representative battlefield is constructed from a blue and red force, each of which consists of a total of six entities. While all entities correspond to generic tanks, five are assumed to form the lowest hierarchy of the command, subordinates to the leader, the sixth unit. Each entity is modeled in the simulation as an autonomous, concurrent, process. The processes are executed on the a mix of available processor platforms, connected through a 10 Mbit/sec Ethernet, including the Sun sparcstations under SunOS 4.1 and Solaris 2.4 and Intel 486DX2/66- and pentium-based machines under the freely available Linux operating system. Inter-process communication is achieved through TCP/IP.

To ensure a consistent view of the battlefield among the peer entities, despite the corresponding autonomous, concurrent processes, each unit maintains a point-to-point connection with its peer and leader and two "control" nodes are introduced into the simulation corresponding to the Blue and Red forces. A control node is merely an artifact of the simulation and has no counterpart in the real world. It maintains an accurate picture of the entire system and propagates information to entities on a need to know basis. Whenever the status of an entity changes, the control node requires an automatic update. Relative to positional information, only the change in the velocity vector needs to be propagated since the control node is capable of computing the final position. In turn, upon request from an entity, whose field of view is limited by its sensor, a control node propagates the corresponding subset of the view of the entire battlefield. For convenience, the entire battlefield is organized into square sectors. Using the knowledge of its sensor capabilities, an entity specifies its sector sect, i.e., the set of sectors in its area of interest of the entire battlefield. Figure 3 presents the organization of the battlefield. For a comparative evaluation of the centralized versus decentralized paradigms, the Blue force, on the left, implement the decentralized paradigm, while the Red force, on the right, is organized under the centralized paradigm.

Figure 3: Interconnection network for the Blue and Red Forces in the Simulation

The accuracy of the simulation results are governed by the spatial and temporal resolutions, i.e., the smallest changes in the location and time that may occur atomically. The finer the granularity, the more accurate the simulation at the cost of increased simulation time. In this paper, the spatial and temporal units are set at 0.2 meters and 0.1 second respectively so that the weapon travel times and communication delays are registered through multiple timesteps. The battlefield is define as a flat terrain, 2,500,000 unit square. Each entity is a generic tank type with the characteristics detailed in Table 1. At simulation initiation, the initial locations of the entities are generated randomly so as to fall within squares of length 15,000 spatial units. Furthermore, the leader units are placed further apart from other units so that they are not engaged in firing. The initial sensor axes for the entities are also generated stochastically.

In addition, the simulation models two delays – (i) the leader-to-entity and peer entity-to-peer entity communication delay, and (ii) decision computing delay. For simplicity and without any loss in generality, this paper assign the same value, a positive integer, τ_c , to the leader-to-entity and peer entity-to-peer entity communication delay. The decision computing delay, τ_p , models the computing time needed to minimize the cost function and assumes that the minimization may require several timesteps. While τ_p may equal several timesteps for the centralized paradigm, τ_p is zero if the decision time is less than a single timestep.

The task of minimizing a cost function is a difficult one and the literature reports a number of algorithms that require reasonable computational costs to generate approximate solutions. However, the goal of this paper is to synthesize a decentralized algorithm under the MFAD framework, corresponding to a centralized decision-making algorithm, and to measure the effectiveness of MFAD.

Characteristic	Value
<i>Movement</i>	
Maximum velocity	15 units/timestep
Maximum acceleration	$\pm\sqrt{2}$ units/timestep ²
<i>Sensor</i>	
Arc size	45°
Rate of change	$\pm 10^\circ$ /timestep
<i>Targeting</i>	
Weapon range	3000 units
Weapon velocity	500 units/timestep
Weapon reload time	25 timesteps

Table 1: Generic Characteristics of Each Entity

Therefore, this paper limits every independent variable in the cost functions to a finite number of integral values and computes exact solutions for both centralized and decentralized paradigms, relatively quickly. Corresponding to the Movement cost function, each entity is only permitted to alter each component velocity vector by ± 1 and the value of the acceleration is assumed constant. The Sensor cost function is expensive to compute, stemming from the desire to minimize sensor overlap between the peers. Conceptually, the problem is equivalent to computing the area of overlap of multiple cones. A simple numerical approximation is chosen to yield rough estimates of sensor overlap, quickly. Also, the maximum change of the sensor axis is limited to 10 degrees and successive increments are limited to 5 degree intervals.

The simulator is written in C and is approximately 18,000 lines long. While it includes both centralized and decentralized implementations, a Motif based graphical interface has been included to allow for both real-time and postmortem viewing of the simulation. Figure 4 presents a screen shot where the small circles represent the individual units and identifiers 2 through 6 and 9 through 13 represent the Blue and Red units, respectively. The arcs originating at each entity reflect the current sensor arc while the short lines of different lengths emanating from the circles represent the magnitude of the current velocity vector.

6 Comparative Performance Analysis of the Centralized and Decentralized Paradigms, under MFAD

To evaluate the effectiveness of the MFAD framework, this section reports the results from executing the centralized and decentralized command and control algorithms, corresponding to the representative battlefield. A number of experiments are designed, as described subsequently.

In every experiment, a number of simulation runs are executed corresponding

Figure 4: A Screen Shot of the Postmortem Replace Interface

to a number of different initial locations of the entities, generated stochastically. In addition, for every experiment, two sets of simulation runs are executed – one with the Red force centralized and Blue force decentralized and another with the Red force decentralized and Blue force centralized. The aim is to eliminate any bias resulting from geographic advantages. Every simulation is executed for a total duration of 600 timesteps which allows sufficient time for the two forces to engage, eventually leaving only one of the two forces with the surviving entities. Both of the paradigms are assigned identical units with identical weapons, similar initial geographic locations for the entities, and identical number of units. The parameter, τ_p , which normally provides an advantage for the decentralized paradigm, is assumed to be zero. Thus, the key remaining factor to discriminate between the two paradigms in MFAD is the communication delay, τ_c , which assumes the form of the independent variable and is assigned values ranging from 1 to 9 timesteps, i.e., 0.1 sec to 0.9 sec, that reflect relatively large communication distances in representative battlefields. The parameter `INFORM_RANGE` is set to 30,000 spatial units.

A set of five measures are obtained and reported in this paper: (i) the average number of enemy units destroyed, by each of the decentralized and centralized paradigms, (ii) sensor and movement error measures for the two paradigms relative to the perfect decision vector, (iii) average sensor error over all entities, as a function of the communication delay, and (iv) average movement error over all entities, as a function of the communication delay, and (v) quantitative

Figure 5: Number of enemy units destroyed under the centralized and decentralized paradigms, as a function of τ_c

evaluation of the communications requirements imposed by the two paradigms.

Figure 5 presents the first measure, namely the average number of enemy units destroyed, by each of the decentralized and centralized paradigms, as a function of τ_c . For the entire range of τ_c and for the large number of initial locations of the Blue and Red forces, Figure 5 reveals that the decentralized paradigm is superior to the centralized scheme. For the lowest delay value, $\tau_c = 1$ timestep, the decentralized paradigm exhibits an advantage of 0.9. The advantage increases to 3.667 for $\tau_c = 9$ timesteps, i.e., 5 enemy units are destroyed by the force utilizing the decentralized paradigm for a loss of only 1.333 of its units.

While the results in Figure 5 clearly demonstrate the performance superiority of the distributed over the centralized paradigm, they lack insight into the quality of the decisions under the distributed paradigm. While the underlying algorithms of the decision-making entity in the centralized paradigm has access to the entire system state, the combination of (i) latency of the information received from its subordinates, (ii) the delay resulting from the significant computational need, and (iii) the latency of execution of the decisions, hinder the generation of high quality decisions. In contrast, while the decentralized paradigm successfully addresses the issues (ii) and (iii), the lack of access to the total system state, limits the quality of its decisions. To provide insight into the quality of the decisions, this paper proposes a hypothetical entity, Perfect Global Optimization Device (PGOD) that is capable of generating perfect decisions. By definition, PGOD transcends time in that it is capable of acquiring any and all knowledge at any spatial location in the system instantaneously, i.e., unaffected by the finite rate of propagation of information that characterizes the physical world. Thus, PGOD utilizes the perfect information and generates perfect decisions and while they may never be realized in the real world, they serve

as the absolute standard for comparing the effectiveness of the distributed and centralized paradigms.

A series of experiments are design with the Blue and Red forces implementing the decentralized and centralized paradigms. The decision vectors computed by each of the “ n ” entities at every timestep are logged, and the overall decision vector for the entire system is expressed through $UD(t)$,

$$UD(t) = \sum_{i=1}^n UD^i(t) = \sum_{i=1}^n [UD_m^i(t), UD_s^i(t), UD_t^i(t)]$$

While the overall decision vector for the entire system for the centralized paradigm may be obtained from the leader of the Red force, the sequence of decisions for the two paradigms are likely to be different, given the asynchronous nature of the decentralized paradigm. For consistency in comparing the decision vectors from the two paradigms, this paper utilizes the sighting reports sent to the leader of the Blue force by the friendly units to compute the corresponding centralized decision, $UC(t)$, at every timestep. However, in a true centralized paradigm, the decisions generated by the leader must be propagated to the individual entities for execution and, therefore, the corrected overall decision vector is given by, $UC(t = T - \tau_c)$, where the delay from the leader to each of the entities is τ_c timesteps.

For the hypothetical entity, PGOD, located at the Blue leader, the state vectors of every entity and the sighting reports from the entities are assumed to be available to it instantaneously at time instant t , i.e., without incurring any propagation delay. PGOD employs the same algorithm as the centralized paradigm and computes the decision vector for each entity. These vectors are considered ideal because they are not subject to the latency inherent in the centralized paradigm and the decision process utilized PGOD’s knowledge of the total system state unlike the decentralized paradigm. The perfect decision vector is labeled $UI(t)$.

The second set of measures in this paper consists of error measures for each of centralized ($E_c(t)$), corrected centralized ($E_{cc}(t)$), and decentralized ($E_d(t)$) decision vectors, relative to the perfect decision vector, as presented through the expressions (11), (12), and (13) respectively.

$$E_c(t) = \sum_{i=1}^n E_c^i(t) \quad \text{where } E_c^i(t) = |UI^i(t) - UC^i(t)| \quad (11)$$

$$E_{cc}(t) = \sum_{i=1}^n E_{cc}^i(t) \quad \text{where } E_{cc}^i(t) = |UI^i(t) - UC^i(t - \tau_c)| \quad (12)$$

$$E_d(t) = \sum_{i=1}^n E_d^i(t) \quad \text{where } E_d^i(t) = |UI^i(t) - UD^i(t)| \quad (13)$$

Figure 6 presents the sensor decision vector for an entity computed under each of the centralized and distributed paradigms and by PGOD, as a function

Figure 6: Sensor decision vector for an entity under the centralized and distributed paradigms and PGOD, as a function of time, for $\tau_c = 1$ timestep

of time, between 15 and 500 timesteps. The value of τ_c is set at 1 timestep. Figure 6 reveals that the graphs corresponding to $UD^i(t)$ and $UC^i(t)$ closely follow that of PGOD. Thus, the sensor decision vector for the decentralized paradigm closely tracks the perfect decision vector. The graph for the corrected centralized paradigm may be obtained by shifting the graph for $UC^i(t)$, shown in Figure 6, by one timestep, and would differ from that of PGOD.

As the value for τ_c increases from 5 timesteps to 9 timesteps, shown through Figures 7 to 8, the $UC^i(t)$ graph lags the $UI^i(t)$ graph considerably. The corrected centralized graphs that may be obtained through shifting the $UC^i(t)$ graph by 5 and 9 timesteps respectively, would differ substantially from that of $UI^i(t)$. That is, the decision vector corresponding to the corrected centralized paradigm differs significantly from that from PGOD. In contrast, the decision vector graph corresponding to the decentralized paradigm, closely follows that of PGOD for large shifts in the sensor axis and differs only slightly for small and rapid changes in the sensor axis. A likely cause of this behavior is the relatively high communication delay value.

The data in Figures 6 through 8 are replotted in 9 through 11 to present the sensor error measure for the centralized, corrected centralized, and distributed scenarios, also as a function of time. The graphs reveal that the error for the corrected centralized is larger than that for the decentralized paradigm and that the magnitude of the error margin increases with increasing values for τ_c .

Figure 12 presents the third measure – average sensor error over all entities, as a function of the communication delay, τ_c . The average error for the decentralized paradigm is observed to be relatively uniform, implying that the decentralized decisions closely track the perfect decisions. In contrast, the quality of decisions in the centralized paradigm deteriorate with increasing values of the communication delay.

Figure 7: Sensor decision vector for an entity under the centralized and distributed paradigms and PGOD, as a function of time, for $\tau_c = 5$ timesteps

Figure 8: Sensor decision vector for an entity under the centralized and distributed paradigms and PGOD, as a function of time, for $\tau_c = 9$ timesteps

Figure 9: Sensor error measures for an entity for centralized, corrected centralized, and distributed scenarios, as a function of time, for $\tau_c = 1$ timestep

Figure 10: Sensor error measures for an entity for centralized, corrected centralized, and distributed scenarios, as a function of time, for $\tau_c = 5$ timesteps

Figure 11: Sensor error measures for centralized, corrected centralized, and distributed scenarios, as a function of time, for $\tau_c = 9$ timesteps

Figure 12: Average sensor error over all entities for centralized, corrected centralized, and distributed scenarios, as a function of τ_c

Figure 13: Average movement error over all entities for centralized, corrected centralized, and distributed scenarios, as a function of τ_c

Figure 13 presents the fourth measure – average movement error over all entities, as a function of τ_c . The behaviors of the corrected centralized and distributed paradigm graphs are similar to that in Figure 13, further confirming the observation that the decentralized decisions closely track the perfect decisions.

Given that an integral component of the distributed paradigm is increased communications, the fifth measure aims at a quantitative evaluation of the communications. Measurements from the simulation indicate that the average number of bytes on the communications links are 289 and 292 per timestep for the decentralized and centralized paradigms respectively. These correspond to link throughputs of 2.89 and 2.92 Kbytes/sec respectively. The maximum throughputs required on any link in the decentralized and centralized paradigms are 16.08 Kbytes/sec and 21.44 Kbytes/sec, respectively. Thus, despite requiring additional communication links, the decentralized algorithm does not impose a higher average link throughput. The key reason is that, unlike the centralized algorithm where the communications is concentrated around the centralized leader, the communications subtask is distributed among all the links in the decentralized algorithm.

The results reported through the graphs up to Figure 13 assumed a constant value of 30,000 spatial units for the parameter, `INFORM_RANGE`. When an enemy unit is sighted within `INFORM_RANGE` of the last known position of a friendly unit i , the sighting friendly unit j will propagate its sighting information to unit i . Therefore, as the value of the parameter, `INFORM_RANGE`, is decreased, the effective area of interest of the battlefield diminishes and the communications are likely to decrease. However, the limited visual field of the battlefield is expected to increase the error. A set of experiments are designed wherein the value of `INFORM_RANGE` is decreased from 30,000 to 15,000 and 5,000 spatial units. For the decentralized paradigm, while the required

average link capacities decrease from 2.89 Kbytes/sec to 2.71 Kbytes/sec and 1.94 Kbytes/sec, the errors increase from 0.143126 to 0.140849 and 0.223081. Even at a relatively low value of `INFORM_RANGE` = 5,000 timesteps, for $\tau_c = 5$, the error for the decentralized algorithm supersedes the error of 0.2609 for the centralized algorithm.

As indicated earlier, the value of τ_p is set to 0 for all of the experiments in order to focus on the impact of communications delay on the quality of decisions. The assumption implies that the time to compute the cost function corresponding to a timestep is completed in less than a single timestep. Such an assumption may be realistic for the decentralized algorithm wherein the overall decisions are shared among the concurrent entities. However, for the centralized algorithm wherein the leader unit solely executes the cost function, the assumption may be inappropriate, particularly where the leader unit has to compute the decisions for a large number of subordinate entities. A non-zero value for τ_p would effectively require the graph for the corrected centralized algorithm to be shifted even further right by τ_p , thereby increasing the error. An informal estimate of the computational costs in the simulations, utilizing the “Unix ps” command, reveals the following. For a total of 600 timesteps of the simulation run, the ratio of the computational time required by the centralized leader unit to that for each of the entities in the decentralized paradigm, is 15:1.

7 Conclusions

This paper has proposed a mathematical framework, MFAD, based on the Kohn-Nerode distributed hybrid control paradigm, to describe a centralized decision-making algorithm and to synthesize from it a distributed decision-making algorithm. Ideally, the centralized control gathers all data from all entities in the system and utilizes them to arrive at the decisions and, as a result, the decisions are expected to be “globally” optimal. In truth, however, as the frequency of the sensor data increases and the environment gets larger, dynamic, and more complex, the decisions come into question. In the distributed decision-making algorithm, the centralized decision-making is replaced by those of the constituent entities that aim at minimizing a Lagrangian, i.e., a local, non-negative cost criterion, subject to the constraints imposed by the global goal. Thus, computations are carried out locally, utilizing locally obtained data and appropriate information that is propagated from other sites. This paper has implemented both the centralized and distributed paradigms for a representative military battlefield and simulated them on a testbed of network of workstations for a comparative performance evaluation of the centralized and decentralized paradigms in the MFAD framework. Performance results have indicated that the decentralized approach consistently outperforms the centralized approach. This paper has also introduced a fundamental concept, embodied through a fictitious entity termed “Perfect Global Optimization Device (PGOD),” towards a quantitative evaluation of the quality of decisions under the decentralized paradigm. PGOD possesses perfect knowledge, i.e., the state information of every

entity of the entire system unaffected by time and delay, at all times. PGOD is therefore capable of generating perfect globally-optimal decisions which, though unattainable, provide a fundamental and absolute basis for comparing the quality of decisions. Simulation results have shown that the quality of decisions in the decentralized paradigm are superior to those of the centralized approach and that they closely track PGOD's decisions.

The perfect or ideas decisions constitute a fundamental property of decentralized algorithms and the authors are currently engaged in understanding its relationship with other fundamental properties including stability.

References

- [1] Raj V. Iyer and Sumit Ghosh, "DARYN, A Distributed Decision-Making Algorithm for Railway Networks: Modeling and Simulation," *IEEE Transactions on Vehicular Technology*, Vol. 44, No. 1, February 1995, pp. 180–191.
- [2] Noppant Utamaphethai and Sumit Ghosh, "DICAF, A High Performance, Distributed, Scalable Architecture for IVHS Utilizing a Continuous Function – 'Congestion Measure'" Accepted for presentation and publication in the proceedings of ITS America Sixth Annual Meeting & Exposition, April 15–18, 1996, George R. Brown Convention Center, Houston, Texas, Organized by the Intelligent Transportation Society of America, Washington, DC 20024.
- [3] Sumit Ghosh, Kwun Han, Ramana Reddy, Sumitra Reddy, Srin Kankanhalli, Juggy Jagannathan, & Bob Shank, "A Distributed, Scalable, Community Care Network Architecture for Real-Time Access to Geographically-Dispersed Patient Medical Records: Modeling and Simulation," Accepted for publication in the Proceedings of the 19th Annual Symposium on Computer Care Applications in Medical Care (SCAMC '95), Oct 28 – Nov 1, 1995, New Orleans, American Medical Informatics Association, DC, pp. 352–356.
- [4] Ling-Rong Chen and Sumit Ghosh, "Modeling and Simulation of a Hierarchical, Distributed, Dynamic Inventory Management (HDDI) Scheme," Submitted for possible publication in *Journal on Computing*, Institute for Operations Research and the Management Sciences (INFORMS), March 1996.
- [5] W. Kohn and A. Nerode, Multiple Agent Hybrid Control Architecture, in *Hybrid Systems*, Lecture Notes in Computer Science, Eds. R.L. Grossman, A. Nerode, A. Ravn, and H. Rischel, LNCS-736, Springer-Verlag, 1993.
- [6] W. Kohn and A. Nerode, A Hybrid Systems Architecture, in *Logical Methods* (J.N. Crossley, J.B. Remmel, R.A. Shore, Moss E. Sweedler, eds.), Birkhauser, 1993.

- [7] H.G. Rotithor, "Enhanced Bayesian Decision Model for Decentralized Decision Making in a Dynamic Environment," Proceedings of the IEEE International Conference on System, Man, and Cybernetics, Vol. 3, 1991, pp. 2091–2096.
- [8] J.N. Tsitsikilis and M. Athans, "Convergence and Asymptotic Agreement in Distributed Decision Problem," IEEE Transactions on Automatic Control, Vol. AC-29, No. 1, January 1984, pp. 42–50.
- [9] J.N. Tsitsikilis and M. Athans, "On the Complexity of Decentralized Decision Making and Detection Problems," IEEE Transactions on Automatic Control, Vol. AC-30, No. 5, May 1985, pp. 440–446.
- [10] W. Kohn, A. Nerode, and J.B. Remmel, "Hybrid Systems as Finsler Manifolds: Finite State Control as Approximations to Connections," Technical Report 95-22, June 1995, Mathematical Sciences Institute, Cornell University, New York.
- [11] G.B. Thomas and R.L. Finney, Calculus and Analytic Geometry, Addison-Wesley Publishing Company, New York, 1985.
- [12] V.L. Volkovich, "Algorithm for solving a linear optimization problem in a distributed system," Soviet Journal of Automation and Information Sciences, Vol. 22, No. 1, 1990, pp. 40–49.
- [13] J.A. Stankovic, "An application of Bayesian Decision Theory to Decentralized Control of Job Scheduling," IEEE Transactions on Computers, Vol. C-34, No. 2, Feb 1985, pp. 117–130.