The Link Foundation Fellowship In Advanced Simulation and Training

The Design and Evaluation of Distributed Graph-Theoretic Heuristics for Computing Pairwise Interactions of Moving Entities

by Darren R. Law

Department of Computer Science
University of Central Florida

September 1998

The
Link Foundation
Fellowship
In Advanced
Simulation and Training

The Design and Evaluation of Distributed Graph-Theoretic Heuristics for Computing Pairwise Interactions of Moving Entities

by Darren R. Law

Department of Computer Science
University of Central Florida

September 1998

The Link Foundation Fellowship In Advanced Simulation and Training

The Design and Evaluation of Distributed Graph-Theoretic Heuristics for Computing Pairwise Interactions of Moving Entities

by Darren R. Law

Department of Computer Science
University of Central Florida

September 1998

The Link Foundation Fellowship in Advanced Simulation and Training

A grant is awarded to foster advanced level study in simulation and training research: to enhance and expand the theoretical and practical knowledge of how to train the operators and users of complex systems and how to simulate the real-world environments in which they function; and to disseminate the results of that research through lectures, seminars and publications. This publication is the final report of the fellow's research.

LINK AWARDEES

1991-92	Joe Dumas / University of Central Florida
1992-93	Julia Carrington / University of Central Florida
1992-93	Ginger Watson-Papelis / University of Iowa
1993-94	Michael Bajura / North Carolina University
1993-94	Eric Foxlin / Massachusetts Institute of Technology
1993-94	Gary R. Geroge / Binghamton University
1993-94	Matthew P. Reed / University of Michigan
1993-94	Benjamin A. Watson / Georgia Institute of Technology
1993-94	Ronald J. Guckenberger / University of Central Florida
1993-94	Henry A. Nelson / University of Central Florida
1994-95	Kurt M. Joseph / Kansas State University
1994-95	Mark R. Mine / University of North Carolina
1994-95	James R. Otto / University of Kentucky
1994-95	Jim Chen / University of Central Florida
1995-96	Ellen Bass / Georgia Institute of Technology
1995-96	Jonathan Cohen / University of North Carolina
1995-96	Doreen Comerford / Kansas State University
1995-96	Curtis Lisle / University of Central Florida
1995-96	Hans Weber / University of North Carolina
1995-96	Donna Wilt / Florida Institute of Technology
1996-97	Robert A. Glaser / University of Massachusetts
1996-97	Leslie L. Heimenz / Ohio State University
1996-97	Mark C. Kilby / University of Central Florida
1996-97	William R. Mark / University of North Carolina
1996-97	Ellen Scher Zagier / University of North Carolina
1996-97	Douglas A. Peterson / University of South Dakota
1997-98	Robert Franceschini / University of Central Florida
1997-98	Barry Kelly / Binghamton University
1997-98	Darren Law / University of Central Florida

1997-98	Mark Livingston / University of North Carolina
1997-98	Glenn Martin / University of Central Florida
1997-98	Carol Chensey / University of Florida

The Institute for Simulation and Training and the University of Central Florida
Administer this program on behalf of the Link Foundation.
Dr. A. Louis Medin, Executive Director
Institute for Simulation and Training
3280 Progress Drive
Orlando, Florida 32826-0544

LINK FOUNDATION FELLOWSHIP REPORT

THE DESIGN AND EVALUATION OF DISTRIBUTED GRAPH-THEORETIC HEURISTICS FOR COMPUTING PAIRWISE INTERACTIONS OF MOVING ENTITIES

SEPTEMBER 30, 1998

The Link Foundation Fellowships in Advanced Simulation & Training

FOR: Link Foundation, c/o Institute for Simulation and Training
3280 Progress Drive
Orlando, FL 32826-0544

BY: Darren R. Law
352 Royal Palm Drive
Melbourne, FL 32935-6955

TABLE OF CONTENTS

TABLE OF FIGURES

1. Introduction

In recent years the focus of training simulations, particularly those in the defense domain, has been on increasing the number of simulated entities. This focus, coupled with the requirement for geographically-distributed training sites, has led to increased interest in distributed simulation of the battlefield environment. The large efforts involved in the development of the DIS (Distributed Interactive Simulation) protocol and the HLA (High Level Architecture) have attempted to address these DoD (Department of Defense) requirements in several ways.

The cost of computing the interactions between pairs of simulated entities frequently dominates system execution time. The difficulty of evaluating these interactions efficiently has been a major impediment to the development of the causality-preserving large-scale human-in-the-loop simulations which the defense training community desires.

Many of the problems which arise in the computation of entity interactions are intractable, and admit no reasonably-efficient optimal solution. This intractability suggests that it is reasonable to seek heuristic solutions to these problems which provide good efficiency for most problem instances. The research described in this report attempted to find distributed graph-theoretic heuristics which would offer improvements in efficiency by simply making judicious mappings of simulated entities to simulation nodes.

2. Problem Description

In many training simulations, some collection of N entities of various types (tanks, infantry, helicopters, etc..) moves through a simulated terrain. For the simulation to be interesting, these entities must interact with each other in some way. If there is no interaction among entities, the same cognitive purpose could be served by running N separate single-entity simulations. Typically, battlefield entities interact in some way, e.g. by detecting, pursuing, avoiding, or attacking one another. In most cases, these interactions are either between two single entities (a helicopter fires on a tank) or between two aggregated groups of entities (a tank platoon sees an infantry company). From these two examples, we can see that the geographic proximity of entities may play an important role in these interactions.

In the latter example above, the interaction was described as being between two groups of entities. In a constructive simulation, this characterization might be acceptable, since the levels of resolution for the two types of battlefield entities given might be the units platoon and company, respectively. In this case the "number of battlefield entities", N, would be the number of units simulated at the finest level of granularity. In a virtual simulation, each of these groups might actually be modeled as a collection of independent entities; in this case, the characterization of the interaction as a single interaction is incorrect. The interaction between the two groups of entities is actually a collection of detections by individual entities, communications about that information, and possibly some reasoning which enables the individual entities representing the tanks (or their personnel) to deduce that the detected individual combatants constitute an infantry company. N is still an accurate description of the number of battlefield entities, but the interaction described is actually a collection of interactions.

It is important to note that each of the interactions in this collection may be reasonably viewed as an interaction between two entities – a tank detects an infantryman or a tank communicates with another tank. Even though the sending of a radio transmission is a single event in reality, that event must be modeled in code by distributing that information to each simulated entity which can detect it, even to those entities who were not intended to receive it. It is difficult to identify interactions among entities which do not have to be modeled as interactions between pairs of entities in software.

In order to provide a view of the simulated world which provides for fairness of interactions and temporally and spatially correct cause an effect, some time synchronization must occur among the nodes of a distributed simulation. This use of simple time-stepped simulation is more amenable to synchronization than a purely event-driven simulation. The use of time-stepped simulations is commonplace in the defense training community.

2.1 Tested Abstraction of the Problem

It is clear that there may be many types of entities and many types of interactions among them. A simplified simulation which abstracts the important characteristics of entity-oriented, time-stepped distributed simulations was developed. This simulation has the following characteristics:

1. There is some fixed number, N, of simulated entities. Each entity has a position, speed, and heading in the xy-plane.
2. Each entity is one of several parameterized entity types. The parameters of an entity type are its speed, its detection range, and its behavior towards other entity types. Various entity types either pursue, avoid, or ignore other types; no pursuit or evasion more advanced than simply changing course to point at or way from another entity is modeled.
3. The simulation runs for a pre-specified number T of time steps.
4. In each time step, the following actions are performed:

- A list of Potential Targets of Interest (PTOI) is generated for each entity.
- Each entity checks each entity in its Potential Targets of Interest (PTOI), if an entity is detected (is within the detecting entity's detection range) it is placed in the Targets of Interest (TOI) list.
- Each entity performs some interaction computation in each entity in its TOI. This may include changing course to avoid, pursue, or intercept some entity in that list.
- Each entity moves in the xy-plane.

Entities are neither created nor deleted in the course of a simulation execution. Only two-dimensional motion in the xy-plane is modeled. The simulated world has pre-defined minimum and maximum x and y extents; if an entity leaves this region, it is simply "wrapped" to the other side of the simulated world to preclude changes in the number of simulated entities which might complicate analysis of experimental results.

2.2 Focus of This Research

The focus of this research is the development of distributed heuristics for the mapping of simulated entities to simulation nodes. It is clearly desirable to balance the computational load at each simulation node by making the numbers of entities simulated by each node as even as possible, but the cost of communicating entity state information between nodes to permit detection and interaction computations must be considered.

3. Solution Approach

The research and experimentation conducted to develop distributed heuristics for the efficient computation of entity interactions differs somewhat from that which was proposed. Early efforts along the lines originally proposed did not yield promising results; consequently, a more general approach was adopted in hopes of achieving efficient computation of entity interactions. The original research proposed and the actual research conducted are described in the following sections.

There are essentially two types of algorithms which can be used to improve overall computation time in distributed entity-based simulations. Balancing algorithms may be applied to efficiently assign entities to processors to minimize inter-processor communication and keep the computational load as evenly distributed as possible. Maintaining a balanced load minimizes the time required by the last simulation node to complete a given step. In conservative simulations like those typical in human-in-the-loop training systems, other nodes cannot proceed to later steps until this last node is done due to the need to preserve the appearance of causality and fairness among nodes. Balancing algorithms may be static, distributing entities among processors only at initialization, or may be dynamic, moving entities among processor nodes after any simulation step. The experiments described in this report all use some form of static balancing in the uninstrumented scenario initialization process.

The second type of algorithm used to improve efficiency in distributed entity-oriented simulations is filtering. Filtering algorithms attempt to reduce network traffic and the cost of processing irrelevant information at simulation nodes by reducing the number of irrelevant data sent to or from each node. The larger success in this research was the development of a very simple distributed filtering algorithm which seems to show reasonable improvements over more naive algorithm.

A third type of algorithm for improving efficiency in distributed simulation makes use of "predictive contracts". In essence, simulated entities distribute information to every node with some guarantee that their behavior will continue for some amount of time; simulation nodes may then maintain a local simulation of that entity to determine whether or not more information is needed. The entity may send out updated information whenever it is changing the nature of the contract in some way. This type of approach places some additional computational burden on simulation nodes in exchange for a reduction in network traffic and communications delays; a well-studied example of this is the use of "dead-reckoning" algorithms in the DIS (Distributed Interactive Simulation) protocol. There is some implicit use of a similar mechanism in the graph-theoretic filtering heuristic described later; in particular, that heuristic relies on the fact that the distance that any entity can move in a single time step is less than the range of any entity's detection capabilities. In general and in scenarios where entity types are very similar, this is a reasonable assumption. If an entity can move farther in a time step than its sensors can detect other entities, that entity would collide with other objects frequently since it would effectively be "driving beyond the range of its headlights".

The rationale for using a graph-theoretic approach to improving the efficiency of entity interaction evaluations is the simple observation that most relationships or interactions among entities are between pairs of entities. The most obvious of these, range, is a symmetric relation. In those cases where the interaction of entities is based on proximity and the interaction range of the two entities is similar, it is clear that the cost of exchanging information between those two entities to permit interaction should be minimized. If we represent entities with nodes of a graph, we can use the adjacency of nodes to represent some proximity relationship or interest relationship. The well-developed field of graph theory appears to offer some promise for efficient entity interaction algorithms and heuristics.

The "Entity-Detection Graph" was used in the research described in this report. This graph simply joins the nodes representing two entities by an edge if the entities are within detection range of one another. To simplify evaluation, the detection range of each entity type was set to 10 times its single-step movement distance.

The chief constraint on heuristics developed under this research is that distributed computation of those heuristics should be simple and reasonably efficient. To this end, the use of global information about the relationships among simulated entities was minimized. The code used by the Parallel Machine Simulator (described later) uses global information to achieve run-time efficiency in the testbed, but the times captured by the testbed are in terms of abstract simulation steps. The actual heuristics used are written in a distributed manner, using only information local to a simulation node or obtained at a time cost across the simulated communications network.

3.1 Description of Research Proposed

The original proposal for this research identified three major tasks:

- A literature review of generalized Ramsey numbers and other potentially-applicable graph invariants.
- The implementation and evaluation of a static sectoring approach to interaction computation on the MasPar MP-1.
- The development, implementation, and evaluation of "efficient distributed heuristics" for the computation of entity interactions on the MP-1.

The research which was conducted differed from that which was proposed in one major respect: implementation and testing of the entity distribution heuristics was conducted not on the MasPar MP-1, but on a simulation of a collection of simulation nodes. This simulation is described in detail in Section 3.2.2.1; the distributed simulation testbed was used in place of the MP-1 for several reasons:

- Arbitrary node connection topologies could be studied.
- Inter-node communication costs could be controlled as experimental parameters.
- Patterns of computation which other than SIMD (Single Instruction Multiple Data) could be tested. This is more in keeping with the trend toward simulations which are distributed rather than parallel

Early in the literature search it became apparent that the utility of generalized Ramsey numbers in determining entity distributions was unlikely to compensate for the difficulty in computing them. The focus of the literature search and heuristic development was shifted to more easily computed and more local properties of the Entity Detection Graph.

3.2 Description of Research Conducted

For the reasons described above, the research actually conducted differed from the plan originally proposed. In the following section the most important results gleaned from the literature search are described; subsequent sections described the experimental testbed and the data and heuristics for load balancing and filtering which were used to evaluate heuristics developed as part of this research.

3.2.1 Summary of Major Literature Search Results

The literature search revealed a number of interesting insights; most importantly, early research made it clear that Ramsey numbers were not as promising as was originally hoped. The literature search also revealed that:

- the problem being addressed is almost certainly NP-hard
- very little of the research which addresses this problem uses a distributed approach
- research is very active in this area, particularly as it pertains to the High Level Architecture (HLA) for distributed simulation developed under the auspices of the Defense Modeling and Simulation Office (DMSO)

3.2.1.1 Ramsey Numbers

The Ramsey number R(p,q) is an integer k such that any graph on k vertices must contain either a clique (every pair adjacent) on p vertices or an independent set (no two vertices adjacent) on q vertices. It is simple to see that R(p,q) = R(q,p), that R(1,p)=1, and that R(2,p) = p for all values of p and q.. The original research proposed was based on the hope that simply knowing either the total number of simulated entities or the number of simulated entities on the local simulation node would permit the development of heuristics which could exploit the derived existential knowledge of a clique or independent set of a certain size. Early efforts along these lines were unsuccessful for several reasons.

The most significant reason is that the values of most Ramsey numbers are not known exactly; the Ramsey numbers which are known exactly (other than those of the form R(1,p) and R(2,p)) is in Table 1 (Radziszowski). A great deal of research is being done simply using various theorems to bound the value of R(p,q) for various larger values of p and q. Since it is so difficult to compute R(p,q), very little research in computing R(p,q) in a highly distributed way is being done. In general, distributed computation is being used as MIMD parallel computation simply to improve search times in this area.

p↓	q→	3	4	5	6	7	8	9
3		6	9	14	18	23	28	36
4		=R(3,4)	18	25	?	?	?	?
5		=R(3,5)	=R(4,5)	?	?	?	?	?

Table 1: The Exactly Known Non-trivial Ramsey Numbers

A second reason that Ramsey numbers did not lend themselves to efficient heuristics is their existential nature: even if it is known, for example, that there is a clique of size 6 in the Entity Detection Graph induced by the entities of a given node, it can require $O(n^6)$ time to identify the nodes of that clique. It is clear that it is efficient in terms of communications to co-locate the elements of that clique, but the CPU time required to identify the clique members is prohibitive. These observations suggest that the use of less existential invariants and subgraph properties is more likely to be fruitful.

3.2.1.2 The NP-Hardness of the Problem

In (Halldorsson and Lau, 1997), an approximation algorithm for partitioning low-degree graphs was described; extensions of that algorithm to the problems of constraint satisfaction, Max Cut, and graph coloring were also described in that research. The "Maximum K-Cut" problem is defined therein as follows:
INSTANCE: Graph G=(V,E), and positive integer k
SOLUTION: A partition of V into k subsets such that the number of edges across subsets is maximized.

The simple problem of determining if a solution to a given instance of Max K-Cut exists, it is clearly NP-hard to find a solution. The relevance of this result is that the Max K-Cut problem is closely related to the problem of minimizing inter-node communications between simulation nodes.

Suppose that we are given an instance of Max K-Cut, given by the graph G=(V,E) and a positive integer k. We can create an Entity Detection Graph $G^C=(V,E^C)$ in time linear in |E|, and let k be the number of simulation nodes. If we can find an assignment of entities to simulation nodes which minimizes the communications between nodes required to perform detections and interactions, that same assignment of vertices to the k subsets in the complementary instance of Max K-Cut maximizes the number of edges crossing subsets. Since the Max K-Cut decision problem is NP-complete, the corresponding complementary search problem of assigning entities to simulation nodes to minimize interprocessor communications is NP-hard, as is the harder problem of minimizing total computation time over all nodes. This informal proof assures us that we can proceed in the development of sub-optimal heuristics confident that no optimal solution can be obtained in the general case in polynomial time.

This polynomial mapping between problems suggests that the work of Halldorsson and Lau should be considered as the basis for entity assignment heuristics for minimizing communications. This attempt was not made, since the search techniques described in that research did not appear to be amenable to distributed implementation.

3.2.1.3 Related Recent Research

Many recent publications address the combined problem of entity distribution and filtering; some of the most recent and/or interesting references are given here simply to direct the interested reader. None of these sources was used in the development of heuristics described in this paper, either due to recency or to a lack of amenability to distribution.

(Steinman 1994) describes the Distribution List Algorithm for parallel proximity detection. This algorithm achieves good performance for moving objects and was implemented in the (Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES) environment.

(Reddy 1998) and (Cohen & Kemkes 1998) were published very recently in the Proceedings of the 1998 Fall Simulation Interoperability Workshop. Reddy uses genetic algorithms to perform filtering in the Distributed Interactive Simulation (DIS) environment. Cohen and Kemkes performed experiments measuring the effectiveness of the Data Distribution Management (DDM) relevance filtering in the HLA Runtime Infrastructure (RTI) version 1.3 using two simulation nodes and various simulation scenarios. Their characterization of federate performance is very similar to the complexity function of the simulation used in this research.

(Stoyenko et. al. 1996) attempts to solve a problem which is very similar in many ways to the simulation problem addressed by this research. Stoyenko attempted to distribute the components of a single program at compilation and uses a run time system to achieve the dual goals of load balancing and communication minimization; this paper is one of the few which attempts to deal with both issues simultaneously.

Stoyenko's approach is not distributed, but the paper mentions several potentially-useful "classical" heuristics (heaviest-edge first, minimal communication, and Kernighan-Lin) which will be examined in future research

Several other sources provided insights into previous efforts related to the problem at hand. (Morse & Steinman 1997) provides an excellent description of the desired characteristics of relevance filtering algorithms in a discussion of Data Distribution Management in the HLA. (Deo, Medidi, and Prasad 1992) addressed load balancing in parallel battlefield simulations. (Hosseini et. al. 1990) describes a distributed load balancing algorithm based on graph edge coloring. (Lang & Deo 1994) provides a very good survey of load balancing and offers a probabilistic model for dynamic load balancing on arbitrary connection topologies. This paper inspired the development and use of the Parallel Machine Simulator described in Section 3.2.2.1.

3.2.2 Experimental Testbed Design

The evaluation and benchmarking process was conducted on an experimental testbed developed for that purpose. This testbed is composed of two main components: the Parallel Machine Simulator, which simulates the behavior of a parallel or distributed collection of simulation nodes, and the Distributed Simulation, an abstraction of an entity-oriented, time-stepped distributed simulation similar to those widely used for military training. Each of these components is highly parameterized; the components and the values of the parameters used in the experiments are described in Sections 3.2.2.1 and 3.2.2.2.

3.2.2.1 The Parallel Machine Simulator

The Parallel Machine Simulator (PMS) is a collection of software modules which emulates the relevant aspects of a collection of interconnecting processing nodes. The number of nodes, interconnection topology of the nodes, and communications latency as a function of hops between processors are all specified in ASCII files which are loaded at run time to instantiate the Parallel Machine. The communications latency function is specified to be either linear with respect to the shortest number of hops between processors or the ceiling of the log base 2 of that value (to represent the effects of efficient switching hardware). The communications latency does not consider bandwidth in any way or simulate the actual routing of information, it bases its computations strictly on the shortest distance between nodes.

The Parallel Machine Simulator was used simply to organize the distributed simulation and provide hardware-related information for data collection purposes. No actual parallel code was written for use on this simulator; in fact, the PMS can be used to represent the performance of any system from a tightly-coupled mesh of parallel processors to the Internet by simply tailoring parameters, topology, and the communications cost function. The PMS was tested and debugged for 4- and 8-connected toroidally wrapped meshes, hypercubes, stars, and fully connected topologies using both the linear and logarithmic cost models. All experiments performed in this research were conducted using a 4-way connected toroidally-wrapped mesh of 64 processors to resemble the MasPar implementation which was originally proposed.

The PMS was written in Borland C++ 4.52 to run under Windows for Workgroups on a 66 MHz Intel 486 CPU. The entire testbed and all test cases written can be executed in 1 MB of memory or less beyond that required by the operating system and Borland development environment.

3.2.2.2 The Distributed Simulation

The Distributed Simulation component of the Experimental Testbed imitates the relevant aspects of a time-stepped, entity-based distributed simulation. The simulation is initialized by loading an ASCII scenario file which describes the two-dimensional extents of the simulation space and the type, initial position, heading, and speed of each entity. The entities are assigned to nodes of the PMS according to the balancing method defined for the particular experiment.

The simulation runs for a pre-specified number of steps, computing the "time" required the scenario to execute those steps in terms of a simple operation count. This count is the sum of the counts required for each of the individual steps. The time required for an individual step is taken to be the greatest time required by any processor to perform the simulated computations and communications prescribed by the following sequence:

- Each simulation node generates a PTOI (Potential Targets Of Interest) list which identifies the set of all entities which might be detectable by some entity local to that node. This is done at the node level so that two entities on a node do not request data about the same remote entity, resulting in unnecessary network traffic. The total number of network hops required to access this data is computed at the node level using the Parallel Machine Simulator's communications cost functions. The total time cost of these communications is multiplied by the experimental constant TXFR to determine the time incurred by the node for communications.
- Each entity checks each entity listed in the PTOI to determine if it is in detection range. Each entity which is detected is placed in the detecting entity's TOI (Targets Of Interest) list. The number of entities checked for detection is multiplied by the experimental parameter TDCK, and the sum of these times for all local entities is added to the node's computation time for that step.
- Each entity interacts with each entity in its TOI; each interaction check incurs a time cost equal to the experimental parameter TEPI. If the types of the interacting entities are appropriate, the detecting entity may modify its heading to point at or away from the nearest entity of particular interest. The average-case cost of this procedure is embedded in TEPI.
- Each entity modifies its position according to its heading and speed, incurring a cost to the node of TESU in the process.

The ordering of this sequence was selected both because it is reasonable from an implementation standpoint and because it does not conflict with the Bulk Synchronous Parallel (BSP) model of computation as described in (Palepu 1998). We can characterize the time required by a simulation node to perform a single simulation step by:

$$T = (N_i * TESU(+ (\Sigma_j D_j * TEPI) + (N_i * Q * TDCK) + (C * TXFR), \text{ where}$$

N_i is the number of entities local to node i,
j ranges over the set of those entities counted by Ni,
D_j is the number of entities within detection range of entity j (also the number of entities in entity j's TOI),
Q is the number of entities in the node's PTOI,
and C is the total number of communications hops to transfer data for construction of the next step's PTOI.
The values TESU, TEPI, TDCK, and TXFR are experimental constants and are normalized so that the least of the constants has value 1.

In all of the experiments conducted in this research, the values given in Table 2 below. The values of the first 3 parameters were determined simply by counting the number of operations used in the test code which performed that function. This suffices from a complexity analysis standpoint, but obviously ignores the vast time difference between an integer addition and a cosine. The TXFR parameter is set to the cost of one mathematical operation; this is almost certainly unrealistically low. This value was deliberately minimized to make Heuristic 0 (entity balancing without filtering) perform as well as it could, making any results about better performance of other methods more credible.

Parameter	Description	Normalized Experimental Value in "Ops"
TESU	Time for Entity State Update	2
TEPI	Time for Entity Pairwise Interaction	4
TDCK	Time for a Detection Check	2
TXFR	Time to transfer Entity State Data 1 hop	1

Table 2: Experimental values for Simulation Time Constants

3.2.3 The Baseline Heuristics

In order to evaluate the distributed heuristics developed as part of this research, the performance of some well-known algorithms over the same set of simulation scenarios was evaluated. These two algorithmic approaches, both well-studied, are described in the subsequent sections.

3.2.3.1 Heuristic 0: Static Entity Balancing Without Filtering

The simplest approach to balancing a computational load among processors is simply to perform a static balancing and hope that the computational load does not shift too much during execution. In the case of an entity-oriented simulation, this can be done simply by distributing the simulated entities among the processors as equally as possible.

This simple approach completely ignores the cost of communicating entity information between processors, and as such can be expected to perform poorly due to the cost of passing information among simulation nodes. This simplest implementation also uses no filtering at all to reduce the transmission of irrelevant information over the communications network; every entity's information must be broadcast to every simulation node. This algorithm can be expected to outperform those distributed simulations which lack even static balancing, but should perform significantly worse than more sophisticated approaches. The only global information required to perform a static load balancing is the number of entities. The implementation tested here ignores the cost of this initial static balancing, since the focus in training simulations is usually on the post-initialization phase of execution.

3.2.3.2 Heuristic 1: Gridded Static Sectoring for Balancing and Filtering

A very simple approach to improving the efficiency of entity interaction computations is static sectoring. There are numerous variations on this approach, but all implementations divide the simulation space into fixed geographic regions and assign each of these regions to some simulation node. The implementation tested in this research divides the simulation space into k non-overlapping, equally-sized rectangles, where k is the number of simulation nodes. Each simulation node is assigned exactly one of these regions. Each simulation node is responsible for simulating all entities which fall within that node's region; this may lead to an imbalance in the computational load among simulation nodes, but provides certain benefits which may offset this. The first benefit is that nearby entities are frequently assigned to the same simulation node, and thus incur no communications cost. The second benefit is that entities which are not co-located but are likely to interact with one another are most probably located on those nodes whose regions are adjacent to that of the interested entity; if regions are assigned to nodes carefully, this property can be used as an implicit filter to minimize communications.

3.2.3.3 Test Cases

Six scenarios were used to evaluate the two baseline heuristics and developed heuristics. These scenarios, the configuration of the Parallel Machine Simulation, and the parameters of the entity types which were used in the experiments are described in the sections which follow.

3.2.3.3.1 Parallel Machine Configuration and Parameters

All experiments were conducted with the Parallel Machine Simulator configured as a 4-connected, toroidally-wrapped mesh with 64 processors. This is essentially identical to a very small version of the MasPar MP-1 which was proposed as the original testbed host, and was chosen for the same reason: its amenability to gridded static sectoring. By using rectangular sectors, adjacent sectors in the simulation space can be assigned to adjacent processors to give the least possible communications cost.

The MasPar MP-1 and many other parallel machines have additional routing hardware which permits communication between processors to grow logarithmically rather than linearly as a function of inter-processor distance. In the experiments conducted to date, the parallel machine's "communication cost model" parameter has been set to linear, not logarithmic.

3.2.3.3.2 Test Scenarios and Simulation Parameters

Six randomly-generated test scenarios with 512 simulated entities each were used to evaluate the various balancing and filtering heuristics. The six scenarios are described in Table 3. The scenarios differ from one another in two different ways. Scenarios 1-3 are composed entirely of entities of entity type 0. Type 0 entities exhibit no behavior which considers the proximity of other entities; while the same interaction costs are accrued by type 0 entities, they completely ignore other entities and gain no benefit from that cost. Each of the entities in scenarios 4-6 is equally likely to be either type 1 or type 2. Type 1 entities always change their heading to point directly at the nearest detected type 2 entity. Type 2 entities always change their heading to point directly away from the nearest detected type 1 entity. Static sectoring and other approaches which fare well when entities are uniformly distributed can be expected to perform better in scenarios 1-3 than in scenarios 4-6.

Scenario	Number of Entities	World Extent	Comments
1	512	1024 x 1024	All entities are type 0
2	512	512 x 512	All entities are type 0
3	512	256 x 256	All entities are type 0
4	512	1024 x 1024	Approximately half of entities are type 1, half are type 2
5	512	512 x 512	Approximately half of entities are type 1, half are type 2
6	512	256 x 256	Approximately half of entities are type 1, half are type 2

Table 3: Test Scenario Descriptions

The second major difference among the six scenarios is entity density; the same number of entities (512) are used in each scenario, but the dimensions of the simulated space in which they move range from 2^{20} down to 2^{16} square units. As entity density increases, the number of successful detections should increase, leading to an increase in time attributable to the costs of entity interactions.

As the extent of the simulated world decreases, static sectoring approaches should continue to improve in performance until such time as the extent of a sector is smaller than entity detection ranges. At that point, an entity could detect other entities on non-adjacent sectors, possibly increasing communications costs to the extent that they offset the benefits of the static sectoring approach. The improvement in performance due to smaller world extent is exactly offset by the increase in entity density in the tested scenarios since the number of entities in adjacent sectors remains constant.

The Distributed Simulation is highly parameterized -- virtually any system parameter of interest can be altered simply by editing the appropriate ASCII file. This flexibility allows the evaluation of a very wide range of situations, but forces the user to add the boilerplate text "with these system parameters" to any results obtained. The two main parameter sets whose values must be specified are those which describe the simulated entities and those which describe the time costs of various computations and communications. The behavior of the 3 entity types used in these experiments was described above; the other relevant entity parameters are given in below, and were the same for entities of each tested type.

Entity Parameter	Value
Speed (in position units per time unit)	1.0
Detection Range (in position units)	10.0

Table 4: Values of Entity Type Parameters Used in Experiments

The maximum and minimum speeds of each entity type were set to be identical for convenience. Detection of one entity by another was a certainty if the entities were within detection range, although this probability could certainly be implemented as a parameter.

4. Baseline Experiments

A total of 4 different balancing and filtering heuristic treatments were tested; heuristics 0 and 1 are as described earlier, static by-entity load balancing without filtering and the use of a gridded static sectoring approach which combines balancing and filtering. Heuristics 2 and 3 both use the "NUTOI" filter to reduce unnecessary computation and network loading and use the same balancing methods used in heuristics 0 and 1, respectively.

Each of the 4 treatments described herein was applied to each of the 6 test scenarios; each simulation was run for only 10 simulation steps. It was initially believed the behavior of the various entity types over time would cause the step time to change as scenarios with various interacting entity types progressed, but multiple runs with larger numbers of simulation steps showed that the time required for each step remained very uniform as simulation time advanced.

4.1 Evaluation of Heuristic 0

Experiment 0 was one of two experiments conducted simply to establish a basis for evaluating the effectiveness of any heuristics developed as part of this research. Experiment 0 uses no filtering mechanisms to limit the exchange of irrelevant data and use only static load balancing to improve efficiency. Simulation entities are allocated uniformly across the simulation nodes with no consideration given to either spatial or behavioral relationships among the entities.

4.1.1 Results and Analysis

The results obtained by applying Heuristic 0 to each of the 6 test scenarios are given in Table 5 below. Each data point represents only one execution of the scenario, since no random events occur in the scenarios tested. The low variations observed in preliminary testing with multiple scenarios generated using parameters identical to those used to generate these 6 test scenarios suggested that only 1 scenario for each parameter set was needed.

Scenario #	Total Scenario Time
1	102616
2	102732
3	103176
4	102560
5	102712
6	103272

Table 5: Heuristic 0 Experimental Results

4.1.2 Conclusions

The average execution time for scenarios 1-3 was 102,841 operations. The average execution time for scenarios 4-6 was only negligibly higher at 102,848. It is reasonable to expect that the variation in performance for scenarios 4-6 would be higher due to the increase or decrease in entity interactions due to type 1 and type 2 entities seeking or avoiding one another.

4.2 Evaluation of Heuristic 1

Experiment 1 was the second of two experiments performed simply to provide a baseline to compare with developed heuristics. The static sectoring approach used divided the simulated space into equal, non-overlapping regions and assigned adjacent regions (up to 8 regions are adjacent either horizontally, vertically, or diagonally) to adjacent processors of a 4-connected, toroidally wrapped mesh topology. This topology is identical to that of the MasPar MP-1, and was chosen since it is ideal for the gridded, static sectoring approach implemented. It is widely known that this approach works well when entities are uniformly distributed throughout a simulation space; for this reason it was expected that Heuristic 1 would perform well in those scenarios where all entities were of type 0.

In the case of nodes on the "boundaries" of the mesh, those regions where artificial wrapping of entity movement might occur were not marked as adjacent, since the wrapping of entity motion is simply a convenience used to keep the total number of entities stable without implementing more intelligent entity behaviors. For example, the node whose sector's northeast boundary was (0,0) was not considered to be adjacent to the node whose southwest corner was (MaxPos,MaxPos). Even though an entity could cross from one sector to another in a single move, that entity was considered to have left the scenario and re-entered at a different point.

4.2.1 Results and Analysis

The results obtained by applying Heuristic 1 to each of the 6 test scenarios are given in Table 5 below. Each data point represents only one execution of the scenario, since no random events occur in the scenarios tested..

Scenario #	Total Scenario Time
1	15909
2	17700
3	18621
4	12465
5	13332
6	15006

Table 6: Heuristic 1 Experimental Results

4.2.2 Conclusions

The average execution time for static sectoring approach in scenarios 1-3 was 17,410 operations, approximately one sixth of the time required by the naive approach of heuristic 0. The average execution time for scenarios 4-6 was even lower, with a value of 13601. This is somewhat surprising, as one might expect that the nonreactive type 0 entities would maintain a more uniform spacing than the collection of hunting and fleeing type 1 and type 2 entities which is present in scenarios 4-6. It is possible that the football-like clustering of entities that might be expected in scenarios 4-6 did not extend across sector boundaries often, since the detection range is relatively small when compared to the sector extents in these scenarios.

The effects of increased entity density in the simulated space are more apparent in experiment 1 than they are in experiment 2. This difference is attributable to the fact that heuristic 1 is responsive to the scenario; heuristic 0 does not incur any additional cost for re-assigning entities as they move and is therefore less responsive to increases in the number of interactions which lead to heading changes.

5. Heuristic Evaluation Experiments

The most successful heuristic developed so far is the "NUTOI" filtering algorithm. The "NUTOI" designation completely describes the filtering process: the set of possibly-detectable non-local entities (Potential Targets Of Interest, or PTOI) for each simulation node is formed from the Neighborhood of the

Union of the Targets Of Interest (TOI) which were previously detected by node-local entities. The initial PTOI is the complete set of remote entities. The viability of this heuristic depends on the satisfaction of two constraints:

- The distance that each of the two entities (observing entity and potentially-observed entity) can travel in a time unit must be less than the detection ranges of the observing entity and its previously-detected neighbors
- The undetectability of the potentially-observed entity in the previous time step must be due to range; clearly if A did not detect C because it was behind a hill within detection range. This is not a problem in the simulation used in these experiments, but must be considered in real applications.

WARNING: This method can lead to failures to detect in-range targets! This can occur if the entity is on or within two motion steps of the 2-dimensional convex hull of the entities on its node, since there might be no elements of the union of TOI's of elements on the node in the direction of an entity which could move within detection range in the next step. In the experiments conducted, a trace of detections achieved indicates that no detections were missed by the NUTOI filter, but there is certainly some possibility that this could occur. The probability of this type of failure is clearly a function of entity density and the ratio of detection range to entity speed.

5.1 Heuristic 2: Static Entity Balancing With NUTOI Filtering

Heuristic 2 simply combines a static by-entity load balancing performed in initialization with the NUTOI filtering algorithm described above.

5.1.1 Results and Analysis

The results obtained by applying Heuristic 2 to each of the 6 test scenarios are given in Table 5 below. Each data point represents only one execution of the scenario, since no random events occur in the scenarios tested..

Scenario #	Total Scenario Time
1	2032
2	2964
3	7050
4	2042
5	2834
6	7033

Table 7: Heuristic 2 Experimental Results

5.1.2 Conclusions

The average execution time for heuristic 2 in scenarios 1-3 was 4,015. This represents nearly a fourfold improvement over the times observed for the static sectoring approach used by heuristic 1 for the same scenarios. The average execution time for scenarios 4-6 was 3,970, a similar improvement over the performance of heuristic 1. It is clear that the NUTOI filtering method is a significant improvement over static sectoring for the set of simulation time parameters, entity type parameters, and parallel machine configuration tested.

It is interesting to note that the improvement of heuristic 2 over heuristic 1 decreases as entity density increases. Further study of this effect may be warranted, since there may be some critical entity density above which static sectoring outperforms NUTOI filtering. Since heuristic 2 does not perform any dynamic entity re-location, it is possible that this decline in relative performance may be due to efficiencies achieved by the co-location of interacting entities which static sectoring achieves.

5.2 Heuristic 3: Gridded Static Sector Balancing With NUTOI Filtering

Heuristic 3 combines the gridded static-sectoring approach to load balancing used by Heuristic 1 with the NUTOI filtering algorithm used in place of the static sector's filtering algorithm (which simply sets the node's PTOI to the union of the entities simulated on all sector-adjacent nodes). In general, the node PTOI generated by the NUTOI method will be a subset of that generated by the static sectoring approach in those cases where the extend of a sector is more than double the average detection range of an entity. The NUTOI filtering code is certainly more difficult to write than the static sectoring code; it also has the disadvantage that it requires a larger set of entity state data (an entity's TOI as opposed to an entity's position) in order to generate the PTOI. The additional cost of sending this larger amount of data is not reflected in the experimental results, since the effects of bandwidth and network congestion on communication times are not considered by the current version of the PMS.

5.2.1 Results and Analysis

The results obtained by applying Heuristic 3 to each of the 6 test scenarios are given in Table 5 below. Each data point represents only one execution of the scenario, since no random events occur in the scenarios tested..

Scenario #	Total Scenario Time
1	4634
2	6091
3	7452
4	3836
5	4385
6	6234

Table 8: Heuristic 3 Experimental Results

5.2.2 Conclusions

The average execution time for scenarios 1-3 was 6,059 operations. The average execution time for scenarios 4-6 was somewhat lower at 4,818. Both of these times represent a significant improvement over the performance of the static sectoring method alone, but it is interesting to note that the combination of static sectoring as a balancing algorithm and NUTOI as a filtering algorithm is less effective than the simpler combination used in heuristic 2. The decline in performance relative to static sectoring as entity density increases is less dramatic than it was in experiment 2, particularly in the scenarios containing the more reactive type 1 and type 2 entities. It is difficult to say whether this is the result of poorer performance in the less entity-dense scenarios or reduced sensitivity of the algorithm to entity density. It seems likely that both are the result of the static-sectored balancing and the communications cost it incurs.

6. Conclusions and Future Research

Several major efforts were conducted in an attempt to develop distributed heuristics to improve the efficiency of evaluating the interactions between entity pairs. The literature search and early development efforts simply confirmed the suspicion that the problem was intractable in terms of optimal solution, that the problem is a topic of current research, and that current knowledge of Ramsey numbers is insufficient to permit their use in efficient distributed heuristics.

The Experimental Testbed developed to evaluate balancing and filtering heuristics is undoubtedly the most valuable product of this research. The flexibility that is provided by the easy re-parameterization of entity types, scenarios, computation times, and the Parallel Machine Simulator makes the Experimental Testbed an excellent tool for evaluating heuristics in a wide variety of simulated conditions.

The NUTOI filtering heuristic, though quite successful in the experiments conducted, must clearly be modified to preclude the possibility of missing detections in order to be of any practical use. Since the possibility of missing detections is a function of an entity's proximity to the convex hull of the entities local to the same simulation node, it is possible that an O(n log n) convex hull computation on each simulation node is part of a solution to this problem.

There is clearly a great deal more research to be conducted on this problem. The correction to the NUTOI filter is an obvious area for improvement. It seems quite likely that some graph-theoretic properties could be used in balancing algorithms. Work on a balancing algorithm which exchanges entities among nodes based on degree in the locally-induced Edge Detection Graph is underway, but was not complete in time for this report.

Another improvement which can be made is the development of a set of scenarios and entity types with parameters drawn from various application domains. The parameters used in these experiments were chosen somewhat arbitrarily; the methods developed on this testbed are only useful to the extent that the test scenarios represent typical application scenarios.

A similar set of parameterizations of the PMS could be developed to represent various realistic network or machine topologies. The PMS could also be improved by simulating message routing and considering data size and bandwidth. The fact that the field of simulating distributed architectures is already quite crowded suggests that making use of a reasonably simple communications model has some merit in terms of benefit achieved for effort expended.

In summary, it is clear that a significant amount of work remains to be done before results derived from this research can be applied. The testbed developed as part of these efforts and the knowledge gained from the literature search will be invaluable in continuing this research

7. Acknowledgments

This research was accomplished with the generous support of the Link Foundation through the award of a Link Fellowship in Advanced Simulation and Training. Without that assistance, this research would certainly be much farther from completion than I consider it to be now. The author wishes to thank the Foundation, its administrators, and its sponsors for supporting the efforts documented herein.

The success of the heuristics developed and tested to date has been limited, but the availability and flexibility of the testbed and the insights gained through experimentation suffice to maintain my optimism as I continue this research.

8. References

Cohen, D., and Kemkes, A., "Applying User-Level Measurements to the RTI 1.3 Release 2," Proceedings of the 1998 Fall Simulation Interoperability Workshop.

Deo, N., Medidi, M., and Prasad, S., "Processor Allocation in Parallel Battlefield Simulation", Proceedings of the 1992 Winter Simulation Conference, pp. 718-725.

Halldorsson, M., and Lau, H., "Low-degree Graph Partitioning via Local Search with Applications to Constraint Satisfaction, Max Cut, and Coloring", Journal of Graph Algorithms and Applications, vol.1, no. 3, pp. 1-13 (1997).

Hosseini, S., Litow, B., Malkawi, M., McPherson, J., and Vairavan, K. "Analysis of a Graph Coloring Based Distributed Load Balancing Algorithm," Journal of Parallel and Distributed Computing, Vol. 10, 1990.

Lang, S., and Deo, N., "Random Load Balancing in Multiprocessor Systems," 1994.

Morse, Katherine and Steinman, Jeffrey. "Data Distribution Management in the HLA : Multidimensional Regions and Physically Correct Filtering".

Palepu, R., "Parallel Computing: Bulk Synchronous Parallel (BSP) Primer", available on the Internet at http://www.scs.carleton.ca/~palepu/bsp_primer.html.

Radziszowski, S., "Small Ramsey Numbers", Dynamic Surveys of Electronic Journal of Combinatorics, available on the Internet at http://www2.xtdl.com/~robg/doc/ramseyBound.html.

Reddy, Y., "Genetic Algorithm Approach for Relevance Filtering in Distributed Interactive Simulation", Proceedings of the 1998 Fall Simulation Interoperability Workshop.

Steinman, J., "Parallel Proximity Detection and the Distribution List Algorithm", Proceedings of the 1994 Electronic Conference on Simulation (ElecSim 1994), available on the Internet at http://scs.org/conference/elecsim94/proxdet

Stoyenko, A., Bosch, J., Aksit,M., and Marlowe, T., "Load Balanced Mapping of Distributed Objects to Minimize Network Communication", Journal of Parallel and Distributed Computing, Vol. 34, 1996, pp. 117-136.