Multi-Objective Optimization of Dead-Reckoning Error Thresholds for Virtual Environments

Jeremy R. Millar, Douglas D. Hodson, Gary B. Lamont, Gilbert L. Peterson
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Wright-Patterson AFB, Ohio 45433

***Abstract*—Design trade-offs between state consistency and system response time are commonplace in virtual environments. Systems typically rely on predictive consistency algorithms such as dead-reckoning to control consistency and response time. Dead-reckoning error threshold selection determines the consistency/response time trade-off. We extend this trade-off space to explicitly account for the concept of system fairness. We derive a multi-objective optimization problem and apply multi-objective evolutionary algorithms to solve for Pareto optimal error thresholds.**

***Keywords*—Virtual simulation environments, dead reckoning, multi-objective optimization**

I. Introduction

In order to maintain the illusion of a shared environment users must view the same information at the same time [1]–[4]. However, waiting for all nodes to acknowledge receipt of a state update has a negative effect on system responsiveness, thus breaking the sense of immersion and reducing the quality of user experience. Indeed, research has shown that network latencies of 60 ms induce enough response lag to detract from the play experience in some networked games, while latencies above 100 ms result in game abandonment [5], [6]. Similarly, high levels of state inconsistency result in user dissatisfaction due to nonsensical results, e.g., dead men shooting [7].

Consistency can be made arbitrarily good at the expense of system responsiveness by executing the system in lockstep across all participating nodes. Responsiveness, on the other hand, is limited by system architecture and network topology [8]. While perfectly responsive systems could in principle achieve perfect consistency, physical constraints such as the speed of light ensure even the most responsive virtual environments will exhibit some inconsistency [9]. Moreover, as responsiveness improves, the scalability of the system is limited as the system infrastructure becomes overloaded with state updates. A primary goal of virtual environment systems design is to maximize system responsiveness while simultaneously maximizing consistency.

An additional design objective, particularly for network games and interactive simulations, is the notion of system fairness. Fairness measures the disparity between nodes with respect to consistency and responsiveness [6]. The more similar the nodes are with respect to these measures, the more fair the virtual environment. Fair environments ensure that all participants experience similar levels of consistency and response and that no user gains an advantage due to system architecture.

Many virtual environments use predictive consistency mechanisms to improve response time to an input while still achieving acceptable consistency. Dead-reckoning [3], [4], [10] is widely used due to its simplicity and performance. Dead reckoning allows for some amount of inconsistency at remote nodes to ensure local response times remain within acceptable perceptual thresholds. This is achieved by allowing remote nodes to predict the state of the local node between state updates. The key parameter controlling response time and consistency is the state update frequency, which is itself determined by an error threshold. This threshold is the system designer's primary means of tuning consistency, responsiveness, and fairness.

In this paper, we characterize the choice of error threshold as a multi-objective optimization problem. Our approach differs from other work in this area by explicitly including fairness as an objective. We solve the resulting tri-objective problem (i.e., consistency, responsiveness, and fairness) using evolutionary algorithms and a simulation of the virtual environment. This approach allows us to choose dead-reckoning error thresholds that are Pareto optimal for a given system architecture. Finally, we use a simulation to explore the consistency, responsiveness, and fairness trade-offs for a simple virtual environment.

The remainder of this paper is structured as follows: Sections II and III briefly discuss the concepts of dead reckoning and fairness, respectively. Section IV presents a multi-objective model of the error threshold problem. Section V outlines a system architecture for solving the multi-objective error threshold problem. Section VI describes experiments undertaken to validate the architecture and compare performance of various optimization algorithms. Section VII analyzes the results of those experiments. Section VIII outlines related work and Section IX concludes.

II. Dead Reckoning

A virtual environment is a distributed software system supporting multiple users interacting in real-time that provides shared senses of space, presence, and time [2]. The IEEE published Standard 1278, Distributed Interactive Simulation (DIS) [10], to provide a common protocol and messaging standard

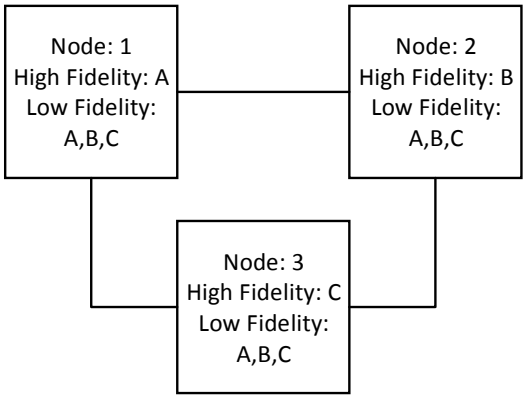


Figure 1. A virtual environment consisting of three nodes and three entities. Each node maintains an authoritative high fidelity model for its local entity and low fidelity models for all entities in the system.

for communicating between nodes in a virtual environment. While there are other standards available (e.g., HLA [11] or TENA [12]), DIS provides a de facto standard for defense oriented virtual environments (i.e., networked simulators) and its consistency maintenance mechanisms are widely used in networked games [4]. Consequently, we restrict our attention to dead-reckoning algorithms as defined by the DIS standard.

The DIS standard defines a predictive consistency maintenance protocol called *dead-reckoning*. Under dead-reckoning, each node maintains a set of low-fidelity models for each remote entity in the system in addition to the high-fidelity models for its hosted entities. Figure 1 depicts a virtual environment consisting of three nodes and three entities. Each node provides an authoritative, high-fidelity model for one entity. Additionally, each node maintains a low-fidelity model of all other entities in the system. Crucially, each node also maintains a low-fidelity model of its own local entity.

The low-fidelity models allow a node to update entity positions between state updates using dead-reckoning algorithms. Low-fidelity models typically operate using simplified dynamics such as first order kinematics. Note that all nodes execute the same dead-reckoning model. State updates are sent by a node whenever the divergence (i.e., difference) between the position of the high-fidelity and low-fidelity models of its hosted entities exceeds a pre-determined threshold. This threshold is the key parameter controlling the dead-reckoning algorithm.

Choosing an appropriate error threshold is a system dependent design decision. Generally speaking, lower thresholds yield better consistency. However, improved consistency comes at the cost of increased network traffic. Depending on network characteristics such as available bandwidth, it is possible to overwhelm the network and increase system response time (that is, the time it takes for all nodes to see an update). Additionally, as network load increases, consistency can actually decrease as well [13].

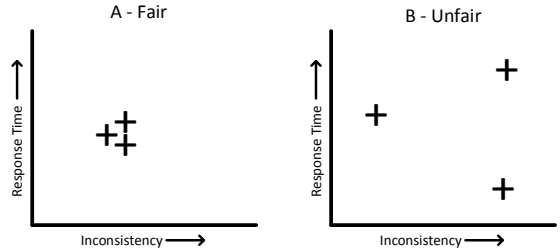


Figure 2. Fairness plots for two three node virtual environments. Each + symbol represents a node's location in two-dimensional fairness space. The nodes in system A are tightly clustered with similar consistency and responsiveness values. Therefore, system A is fair. Conversely, system B is unfair since the nodes are widely dispersed in fairness space.

III. Fairness

Consistency and response time are local properties, that is, they are measured pair-wise. Thus the consistency measured between nodes 1 and 2 with respect to entity A in Figure 1 might well be different than that measured between nodes 2 and 3. Global properties are also of interest, particularly the notion of fairness [6], [14]. A system is fair if no user has an advantage over others due to consistency or response time.

Fairness is measured by projecting each node participating in a virtual environment into a two-dimensional fairness space with consistency as one dimension and system response time as the other. A cluster cohesion measure such as within-class scatter is computed for all nodes. Low scatter indicates that the nodes are tightly clustered in fairness space. Thus, each node has similar consistency levels and response times and no participant experiences a significant advantage or handicap. On the other hand, a high scatter value indicates that nodes have dissimilar consistency values and response times. This affords some participants advantages in terms of state consistency or response while handicapping others.

Figure 2 illustrates these ideas for two three-node virtual environments labeled system A and system B. Note that the horizontal axis measures *inconsistency* so that consistency degrades as one moves away from the origin. The vertical axis measures system response time, i.e., the time required for a state update from one node to reach all other nodes. For both dimensions lower values (closer to the origin) are more desirable. Each '+' symbol indicates an individual node's position in fairness space based on average consistency level and response time.

For system A, the nodes are clustered fairly tightly, indicating a fair system. Each participating node has a similar consistency level and response time. Thus no participant has a distinct advantage in terms of better information about the environment or more rapid environmental response. Conversely, the nodes in system B are widely dispersed in fairness space. This indicates an unfair system. One node has a distinct advantage in terms of data consistency, one has an advantage in response time, and one is severely handicapped in both dimensions.

A system can be fair while exhibiting poor performance with respect to data consistency or response time. Similarly, a system with generally good performance can be unfair so long as at least one node has sufficiently different performance characteristics. Consequently, it is incumbent upon system designers to consider fairness in addition to the more traditional trade-offs between consistency and responsiveness.

IV. Multi-Objective Model

In order to optimize the dead-reckoning error threshold, we need to define and compute the following quantities:

1) average inconsistency,
2) average response time,
3) and fairness.

A. Computing Inconsistency

The virtual environment community has settled on two major inconsistency measures:

1) spatial inconsistency (variously termed spatial error, export error, etc),
2) and time-space inconsistency [15],

Spatial inconsistency is simply the difference between the local, dead-reckoning estimate of an entity's position and its true, high-fidelity position. Time-space inconsistency is the spatial inconsistency integrated over a time period to account for the fact that even small errors can be meaningful if they last long enough. Of the two, spatial inconsistency is the more common, largely because it is simple to compute. Additionally, choosing thresholds for time-space inconsistency can be non-intuitive since the value no longer corresponds to a simple error. For these reasons, we consider spatial inconsistency as our measure of interest for this research. However, the optimization techniques employed here are applicable regardless of the specific inconsistency measure. Indeed, they may well make time-space inconsistency more attractive by eliminating manual input of the threshold value.

We compute the average spatial inconsistency as follows: let $P_i(t)$ be the true position of entity i at time t. Let $P_i^j(t)$ be the position of entity i as represented by node j at time t according to its dead-reckoning model. Then the average (pairwise) inconsistency with respect to entity i at node j is given by

$$\frac{1}{T}\sum_{t=1}^{T}|P_i(t)-P_i^j(t)| \tag{1}$$

Note that this assumes a one-to-one mapping between enitities and hosts; i.e., node i hosts the high-fidelity model for entity i (and no others). Averaging Equation 1 over all entities for a particular node j gives the average spatial inconsistency experienced by node j, i.e.,

$$\frac{1}{N}\sum_{i=1,i\neq j}^{N}\frac{1}{T}\sum_{t=1}^{T}|P_i(t)-P_i^j(t)| \tag{2}$$

Computing Equation 2 for all nodes and averaging provides the average global system inconsistency associated with remote entity positions, i.e.,

$$\frac{1}{N^2T}\sum_{i=1}^{N}\sum_{i=1,i\neq j}^{N}\sum_{t=1}^{T}|P_i(t)-P_i^j(t)| \tag{3}$$

We desire to minimize this inconsistency measure.

B. Computing Response Time

Local response time is associated with the time it takes to process user inputs. We consider the response time associated with propagating state updates from a given node to all other nodes in the system. In order to account for queuing effects in the implemented software system itself, as well as all network-induced latencies, this value should be measured in an end-to-end fashion. That is, the clock begins when the sending application executes the send operation and not when the operating system and network hardware actually place the bits on the wire. Similarly, it ends when the receiving application (not host or operating system) has received the data.

Response time can be calculated as follows: let t_{ij} be the amount of time required to send an update from node i to node j. The the response time is given by

$$\max_j t_{ij}, j=1\ldots N, j\neq i \tag{4}$$

where N is the total number of nodes in the system. Note that this value may vary with time since it depends on environmental factors such as network load. For simplicity, we assume this value is constant.

Averaging Equation 4 over all nodes provides a measure of system response time, i.e.,

$$\frac{1}{N}\sum_{i=1}^{N}\max_j t_{ij}, j=1\ldots N, i\neq j \tag{5}$$

We seek to minimize this response time.

C. Computing Fairness

Equations 2 and 4 provide a means of locating each node in a two-dimensional fairness space. System fairness is computed as the cohesion of the resulting data cluster. Let the vector f_i be the location in fairness space of node i. Then the system fairness is given by

$$\sum_{i=1}^{N}(f_i-c)^2 \tag{6}$$

where c is the centroid of the N fairness locations f_i. Minimizing this value corresponds to a tighter grouping in fairness space.

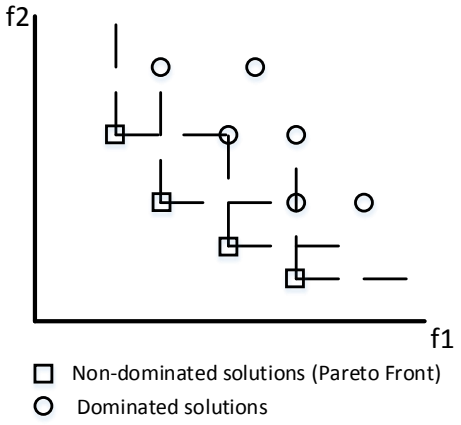


Figure 3. Pareto front, dominated solutions, and non-dominated solutions for a bi-objective minimization problem.

D. Multi-Objective Error Threshold Problem

We are now in a position to define selection of the dead-reckoning error threshold as a multi-objective optimization problem. Let x be the spatial error threshold. Let $\vec{f} = (f_1 f_2 f_3)$, where f_1 is given by Equation 3, f_2 is given by Equation 5, and f_3 is given by Equation 6. Then we wish to find

$$\min_x f(\vec{x}) \text{ s.t. } BW - BW_{max} \leq 0 \tag{7}$$

where BW is the system bandwidth requirement based on the number of state update messages sent and BW_{max} is the system's maximum available bandwidth.

V. System Architecture

In general, there is not a single solution to the multi-objective optimization problem defined by Equation 7. Instead, a set of solutions characterizing the trade-offs between individual objectives is obtained. This notion is formalized through the concepts of Pareto dominance and Pareto optimality.

Definition 1 (Pareto Dominance): Without loss of generality, assume a multi-objective minimization problem. A solution x dominates solution y if $f_i(x) \leq f_i(y)\ \forall i$ and $\exists j$ such that $f_j(x) < f_j(y)$. Pareto dominance is denoted $x \preceq y$.

Definition 2 (Pareto Optimality): A solution x is Pareto optimal if $\neg\exists\ x' \preceq x$; that is, if no other solution dominates x.

A set of Pareto optimal solutions is called a Pareto optimal set and its image in objective space is called the Pareto front. In solving multi-objective optimization problems, we seek to find or approximate the Pareto optimal set and its associated trade-offs represented by the Pareto front.

Figure 3 presents these concepts graphically for a bi-objective minimization problem. Square dots represent non-dominated solutions on the Pareto front. Round dots represent dominated solutions. The dotted lines represent dominance areas – a solution denoted by a square dot dominates any solution above and to its right. Although not drawn, this relationship holds for solutions not on the Pareto front as well.

Solutions to multi-objective optimization problems should lie on or as close as possible to the Pareto Front. Additionally, solutions should cover a broad section of the Pareto front. Multi-objective evolutionary algorithms are a preferred means of solving multi-objective optimization problems because they can find multiple Pareto optimal solutions in a single run. Additionally, multi-objective evolutionary algorithms are able to handle concavity and discontinuity on the Pareto front [16] making them ideal for exploring the trade-off space.

Implementation of a solver for the error threshold problem requires two fundamental subsystems: a multi-objective optimization routine, and a simulation of the virtual environment. We built optimization portion of our solver on the JMetal [17] multi-objective optimization framework. JMetal is a Java-based framework providing abstractions for problems, algorithms, and experiments. It includes a large number of MOEAs as well as standard benchmark problems. Additionally, JMetal provides an experimental framework capable of multiple independent runs and basic statistics gathering. We have extended JMetal with an implementation of the error threshold problem. This extension evaluates candidate solutions by invoking a virtual environment simulator, reading its output, and computing values for each objective function.

A simulation of the distributed virtual environment was developed using the OMNet++ [18] discrete event simulation framework. OMNet++ is a discrete event simulation framework for building C++-based network simulation. Simulations are defined in terms of interacting modules that communicate via timed messages defining the events in the system. Runtime libraries are provided to manage the simulation infrastructure (e.g., the future events list, event scheduling, etc). Extensions provide a variety of network nodes and protocols to assist developers.

For each evaluation, our solver generates a network description file describing the node types, network topology, and parameters to simulate. With the exception of the dead-reckoning error threshold to evaluate, the contents of this file are fixed. The simulator is invoked with the error threshold under consideration and run for a configurable number of time steps. It outputs trajectory and message log files for each entity in the virtual environment.

The trajectory log file for each entity includes its actual position at each time step. It also includes, for each time step, the perceived location of all other entities in the system. Taken as a whole, these data allow us to reconstruct the actual and perceived locations for all pairs of entities at all times.

The message log file for each entity records the start time for each message sent as well as the time each incoming message was received. Taking these data as a whole allows us to compute maximum response times for each state update.

TABLE I
ALGORITHM PARAMETERS.

Parameter	NSGA-II	SPEA2	MCTS
Population size	100	100	100
Archive size	100	100	100
Max evaluations	500	500	500
Crossover probability	0.9	0.9	-
Crossover distribution index	20.0	20.0	-
Mutation probability	1.0	1.0	-
Mutation distribution index	20.0	20.0	-
Exploration coefficient	-	-	$\frac{1}{\sqrt{2}}$

VI. EXPERIMENTS

Validation of the multi-objective approach to setting error thresholds requires a particular virtual environment for investigation. This environment should be deterministic with well understood decision models and dynamics for each entity. Additionally, all entities should be synthetic to allow for statistically significant numbers of trials and long simulations. Finally, the dynamics of each entity should depend on one or more other entities and should be complex enough to provide interesting data.

We leverage Reynolds' boids model of flocking behavior [19] to provide a simple system with complex enough dynamics to generate an interesting Pareto front. The model defines flocking behavior as an emergent system property based on individual behaviors. Entities called boids move through a virtual space in three dimensions. The behavior of each boid is highly coupled to all other boids as each seeks to align its motion with its neighbors, steer towards the center of its neighbors, and avoid collisions. These simple behaviors allow complex flocking to emerge without explicitly designing it into the system.

Boids are attractive as a system model for a number of reasons. Firstly, individual boid behaviors are easy to reason about even though the aggregate flock behavior can be complex. Secondly, boids represent a worst-case scenario in that each entity's behavior depends on all other entities at every time step. Real systems with less coupling should outperform the boids model. Finally, we can define flock cohesion, or the average distance between a boid's flight path and the flock's mean flight path, as a simple measure of system performance.

To investigate the shape of the error threshold Pareto front for the boids, a series of single factor experiments were undertaken. The goals of these experiments are to: 1) demonstrate the validity of the multi-objective optimization approach to determining dead-reckoning error thresholds, 2) ascertain the shape and location of the Pareto front for a representative virtual environment, and 3) compare the performance of the NSGA-II [20], SPEA2 [21], and MCTS [22] multi-objective optimization algorithms on the error threshold problem.

Two experiments were run using slightly different configurations. In the first, a 3 node fully-connected boids network was established. 30 runs were made for each of the NSGA-II, SPEA2, and MCTS algorithms. The simulation ran for 60,000 steps for each candidate solution. All network parameters (e.g., propagation delay, jitter, etc) were fixed and homogeneous. This leads to a constant response time based solely on the network's propagation delay. Therefore, a simple count of messages sent was substituted for Equation 5 with a goal of minimizing total traffic. This is a reasonable thing to do since it models aggregate traffic levels and is associated with system scalability.

TABLE II
WILCOXON RANK-SUM TEST RESULTS FOR EXPERIMENT 1.

	SPEA 2	MCTS
NSGAII	▲	▲
SPEA2		▲

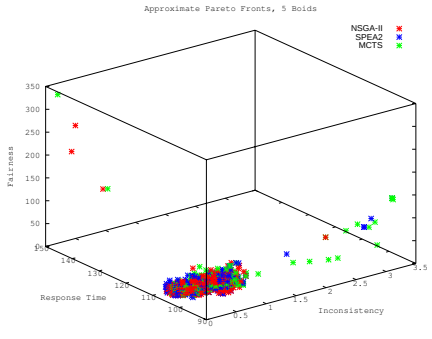
The second experiment used a 5 node fully-connected boids network with time-varying network characteristics. Each link was given a constant propagation delay of 500 ms and a fixed bandwidth of 1.5 Mbps. For each transmission, jitter was sampled from a truncated normal distribution with mean of 100 ms and variance of 60 ms. Link saturation was modeled with a simple queuing mechanism – messages are held until the link becomes available. Retransmits and dropped packets were not modeled. Response time was measured as the second objective. 10 runs were made for each algorithm with 60,000 steps per simulation invocation.

Table I lists the parameters used for each algorithm. Note that crossover, mutation, and selection operators refer to built-in operators provided by JMetal. The exploration coefficient for MCTS sets a trade-off between exploration of new tree branches and exploitation of known good branches. For multi-objective problems, there should be a coefficient for each objective. Since little was known *a priori* about the structure of the search space, all objectives use the same coefficient value.

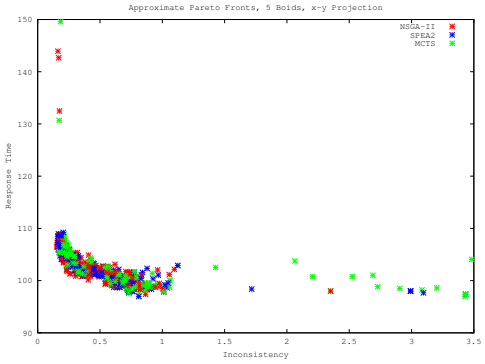
VII. RESULTS AND ANALYSIS

Figure 4 plots the approximate Pareto fronts returned by NSGA-II, SPEA2, and MCTS in objective space for Experiment 1. All three algorithms achieve good convergence and diversity and show a distinct Pareto front. As expected, the best values cluster near the origin. There are well-defined trade-offs between inconsistency and message traffic and inconsistency and fairness. Low volumes of messaging result in high inconsistency and poor fairness.

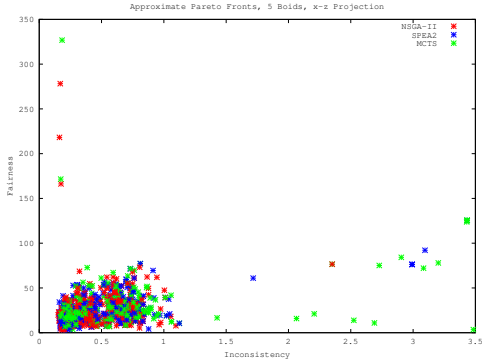
Algorithm performance was compared using the hypervolume quality indicator [16]. The hypervolume indicator measures how much of the objective space is dominated by the solutions in a given set. Consequently, it provides a good indicator of both convergence to the Pareto front and diversity. The hypervolume was calculated for each algorithm run ($N = 30$). Algorithms were compared using the Wilcoxon rank-sum test against the null hypotheses that the median samples were drawn from the same distribution. The Wilcoxon results indicate NSGA-II outperforms both SPEA2 and MCTS. Additionally, one-way ANOVA indicates statistically significant differences in the sample medians ($p = 0.0,\ \alpha = 0.05$).



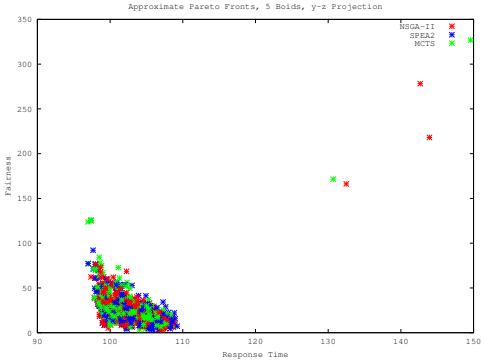
(a) Approximate Pareto fronts achieved by NSGA-II, SPEA2, and MCTS for Experiment 2.



(b) Projection on the inconsistency-response time plane.



(c) Projection front on the inconsistency-fairness plane.



(d) Projection front on the response time-fairness plane.

Figure 5. Approximate Pareto fronts achieved by NSGA-II, SPEA2, and MCTS for Experiment 2.

Figure 5a plots the approximate Pareto fronts returned by NSGA-II, SPEA2, and MCTS in objective space for Experiment 2. Figures 5b to 5d plot the planar projections of the data in Figure 5a. All three algorithms achieve good convergence and diversity and show a distinct Pareto front.

Algorithm performance was again compared with respect to the hypervolume indicator. Due to the small sample size ($N = 10$), the Mann-Whitney rank-sum test was used instead of the Wilcoxon rank-sum test. Results are tabulated in Table III. No statistical difference was found between NSGA-II and SPEA2 or between SPEA2 and MCTS. However, NSGA-II was found to outperform MCTS ($p = 0.0493,\ \alpha = 0.05$).

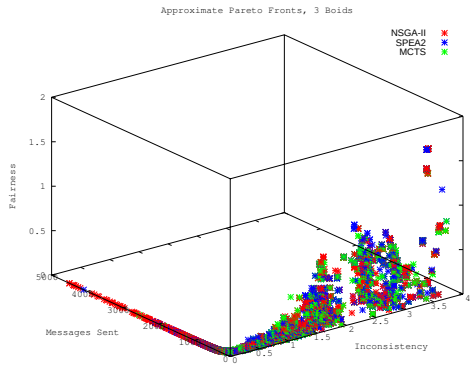


Figure 4. Scatter plot showing the approximate Pareto fronts returned by NSGA-II, SPEA2, and MCTS for Experiment 1.

TABLE III
MANN-WHITNEY RANK-SUM TEST RESULTS FOR EXPERIMENT 2.

	SPEA2	MCTS
NSGA-II	-	▲
SPEA2		-

Both experiments show well-defined Pareto fronts indicating trade-offs between inconsistency, response time, and fairness. Additionally, there are definite lower limits to performance in any of these dimensions.

For example, Experiment 2 clearly shows that there is a minimum achievable inconsistency and that as inconsistency approaches this value, the response time increases dramatically. This drives an attendant degradation in fairness. This result is important since it implies that there are diminishing returns as one approaches the theoretical minimum for inconsistency (see Figures 5b and 5c).

Fortunately, there is a wide area of acceptable performance with low inconsistency, low response time, and reasonable fairness. However, there is wide variability in fairness in this region driven primarily by changing message latency due to jitter and congestion. It should be noted; however, that while response times remain low in this region, the amount of state updates sent becomes rather large as borne out by Experiment 1. The systems under test in this work are small; real-world systems include many more entities and nodes. Thus, while response time may not become a design constraint, overall message volume may well limit scalability. Additionally, while not modeled here, one should also expect response time to increase as message volume increases due to network routing.

A second observation is that the achieved spatial inconsistencies are quite small. The lower bound is approximately 0.25 spatial units. The boids simulation under study uses a virtual world 20 units wide in each dimension, giving an achievable spatial error of about 1%. Whether this is an acceptable level of error depends on the purpose of the simulation. For the boids, velocities are small and 1% error is quite good as the boids maintained their flocking behavior. However, for an aircraft simulation with large velocities (e.g., supersonic), 1% error can translate to a large distance. This too is a valuable result for virtual environment designers – the best achievable objective values may well be too large for the intended application.

Additionally, use of a non-optimal threshold has distinct negative effects on system performance, i.e., flock cohesion. A three boid system with no network delay and perfect information (each entity has the exact position of all others at all times and dead reckoning is not necessary) achieves a flock cohesion measure of 7.667. Adding a modest network delay of 100 ms and an optimal dead reckoning threshold achieves a cohesion measure of 8.573. However, even a slightly non-optimal threshold of 2% spatial error results in increased inconsistency and a flock cohesion measure of 17.45.

Furthermore, conducting a study such as this during the system design phase can bring some insight into fitness of purpose. For instance, in the military domain one might wish to perform an operational test of some weapons system capability using existing simulation resources. If those resources are unable to achieve requisite consistency or response times based on an analysis such as this, the use of simulators should be abandoned. Similar results hold for other domains. In general, this approach can provide the designer some assurance that meeting performance requirements for consistency, response time, and fairness are achievable goals.

Finally, NSGA-II outperformed both SPEA2 and MCTS on the error threshold problem. This was unexpected since SPEA2 and most modern methods tend to outperform NSGA-II by a wide margin on standard test suites. However, the result is not unwelcome since NSGA-II executes faster than both SPEA2 and MCTS.

Some caution in applying these results is in order. It should be noted that the simulation used was a relatively low fidelity network simulation. Bit and packet errors were not modeled. Neither were routing effects, retransmits, or dropped packets.

VIII. RELATED WORK

A common objective in virtual environment research and design is the maintenance of adequate consistency levels in the face of limited system resources such as throughput or network latency [23]. Several authors ([2]–[4], [23], [24]) have highlighted the trade-off between system consistency and system responsiveness as a defining characteristic of virtual environments.

Several papers have been published optimizing various aspects of consistency management and state update scheduling

in particular [25]–[29]. Li and Cai [30] formulate the problem of minimizing inconsistency subject to network capacity constraints as a convex optimization problem. Tang and Zhou [31] derive optimal update schedules based on minimizing time-space inconsistency [15]. However, these analyses do not account for fairness as an explicit objective.

Chen and Zarki [14] define a relationship between system consistency, network delay, and quality of experience, including fairness, but do not provide methods for finding the optimal trade-off point.

IX. Conclusion

Choosing dead-reckoning error thresholds for distributed virtual environments is a non-trivial undertaking. The threshold choice impacts system performance in a number of dimensions including consistency, response time, and fairness. We have presented a multi-objective optimization model that accounts for the relationships between these three quantities. Additionally, we have built an architecture for the associated optimization problem based on multi-objective evolutionary algorithms and simulation of a simple virtual environment.

Experiments with simple virtual environment models indicate use of multi-objective optimization techniques is a viable means of choosing dead-reckoning thresholds. Analysis of the resulting Pareto fronts show diminishing returns as one approaches the theoretical minimum for state inconsistency. Additionally, our results highlight the trade-offs between consistency, response time, and fairness. Application of our model and solver with appropriate entity dynamics can assist virtual environment designers in tuning their applications for best possible performance.

References

[1] S. K. Singhal, "Effective remote modeling in large-scale distributed simulation and visualization environments," Ph.D. dissertation, Stanford University, 1996.

[2] S. Singhal and M. Zyda, *Networked virtual environments: design and implementation*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999.

[3] J. Smed and H. Hakonen, *Algorithms and Networking for Computer Games*. Wiley. com, 2006.

[4] A. Steed and M. F. Oliveira, *Networked Graphics: Building Networked Games and Virtual Environments*. Elsevier, 2009.

[5] P. Quax, P. Monsieurs, W. Lamotte, D. De Vleeschauwer, and N. Degrande, "Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game," in *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. ACM, 2004, pp. 152–156.

[6] J. Brun, F. Safaei, and P. Boustead, "Fairness and playability in online multiplayer games," *Faculty of Informatics-Papers*, p. 232, 2006.

[7] M. Mauve, "How to keep a dead man from shooting," in *Interactive Distributed Multimedia Systems and Telecommunication Services*. Springer, 2000, pp. 199–204.

[8] D. D. Hodson and R. O. Baldwin, "Performance analysis of live-virtual-constructive and distributed virtual simulations: defining requirements in terms of temporal consistency," 2009.

[9] Y. Zhang, L. Chen, and G. Chen, "Globally synchronized dead-reckoning with local lag for continuous distributed multiplayer games," in *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2006, p. 7.

[10] DIS Steering Committee, "IEEE standard for distributed interactive simulation-application protocols," *IEEE Standard*, vol. 1278, 1998.

[11] J. S. Dahmann, R. M. Fujimoto, and R. M. Weatherly, "The department of defense high level architecture," in *Proceedings of the 29th conference on Winter simulation*. IEEE Computer Society, 1997, pp. 142–149.

[12] J. R. Noseworthy, "The test and training enabling architecture (TENA) supporting the decentralized development of distributed applications and lvc simulations," in *Distributed Simulation and Real-Time Applications, 2008. DS-RT 2008. 12th IEEE/ACM International Symposium on*. IEEE, 2008, pp. 259–268.

[13] D. Marshall, S. McLoone, T. Ward, and D. Delaney, "Does reducing packet transmission rates help to improve consistency within distributed interactive applications?" 2006.

[14] P. Chen and M. El Zarki, "Perceptual view inconsistency: an objective evaluation framework for online game quality of experience (qoe)," in *Proceedings of the 10th Annual Workshop on Network and Systems Support for Games*. IEEE Press, 2011, p. 2.

[15] S. Zhou, W. Cai, B.-S. Lee, and S. J. Turner, "Time-space consistency in large-scale distributed virtual environments," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 14, no. 1, pp. 31–47, 2004.

[16] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuisen, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[17] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997811001219

[18] A. Varga, "Omnet++," in *Modeling and Tools for Network Simulation*. Springer, 2010, pp. 35–59.

[19] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 25–34.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[21] E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[22] W. Wang, M. Sebag *et al.*, "Multi-objective monte-carlo tree search," in *Asian conference on machine learning*, 2012.

[23] D. Delaney, T. Ward, and S. McLoone, "On consistency and network latency in distributed interactive applications: A surveypart i," *Presence: Teleoperators and Virtual Environments*, vol. 15, no. 2, pp. 218–234, 2006.

[24] ——, "On consistency and network latency in distributed interactive applications: A survey-part ii," *Presence: Teleoperators and Virtual Environments*, vol. 15, no. 4, pp. 465–482, 2006.

[25] K.-H. Shim and J.-S. Kim, "A dead reckoning algorithm with variable threshold scheme in networked virtual environment," in *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 1113–1118.

[26] Y. Li and W. Cai, "Consistency aware dead reckoning threshold tuning with server assistance in client-server-based dves," in *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 2925–2932.

[27] Z. Li, W. Cai, X. Tang, and S. Zhou, "Dead reckoning-based update scheduling against message loss for improving consistency in dves," in *Principles of Advanced and Distributed Simulation (PADS), 2011 IEEE Workshop on*. IEEE, 2011, pp. 1–9.

[28] Z. Li, X. Tang, W. Cai, and X. Li, "Compensatory dead-reckoning-based update scheduling for distributed virtual environments," *SIMULATION*, 2013.

[29] W. Cai, F. Lee, and L. Chen, "An auto-adaptive dead reckoning algorithm for distributed interactive simulation," in *Proceedings of the thirteenth workshop on Parallel and distributed simulation*. IEEE Computer Society, 1999, pp. 82–89.

[30] Y. Li and W. Cai, "Determining optimal update period for minimizing inconsistency in multi-server distributed virtual environments," in *Proceedings of the 2011 IEEE/ACM 15th International Symposium on Distributed Simulation and Real Time Applications*, ser. DS-RT '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 126–133. [Online]. Available: http://dx.doi.org/10.1109/DS-RT.2011.10

[31] X. Tang and S. Zhou, "Update scheduling for improving consistency in distributed virtual environments," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 6, pp. 765–777, 2010.