ADAPTIVE CONTROL OF LARGE-SCALE SIMULATIONS

A Thesis
Presented to
The Academic Faculty

by

Kirk C. Benson

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
July 2004

ADAPTIVE CONTROL OF LARGE-SCALE SIMULATIONS

ACKNOWLEDGEMENTS

First, and foremost, I would like to acknowledge the contribution of my advisors, Dr. Amy Pritchett and Dr. Dave Goldsman. Both serve as role models in the education and research fields as they inspire those around them to seek knowledge out of sheer enjoyment of the learning process. Thanks to their patience and motivation, this research endeavor has been rewarding on both a personal and professional level. Also, I would like to acknowledge the ISYE faculty at large as they have been insightful and encouraging in all pursuits.

My wife, Shelley, is always supportive and endearing. Her help and humor make all of this possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

Adaptive – modification in reaction to changing circumstances to maintain a desired behavior or effect a desired outcome

ATC – Air Traffic Control

ARTCC – Air Route Traffic Control Center

Configuration – relative arrangement of parts or elements – a defined construct for a model

Control – to exercise restraining or directing influence over

Design – to create, fashion, execute, or construct according to plan – typically an iterative process to achieve defined goals for a system

Embedded – to make something an integral part of

Metric – a standard of measurement

Model – an example for imitation or emulation – set of logical relationships defining a system

LP – Logical Process

NAS – National Airspace System

NOW – Network of Workstations

PDS – Parallel and Distributed Simulation

RFS – Reconfigurable Flight Simulator

RS – Ranking and Selection

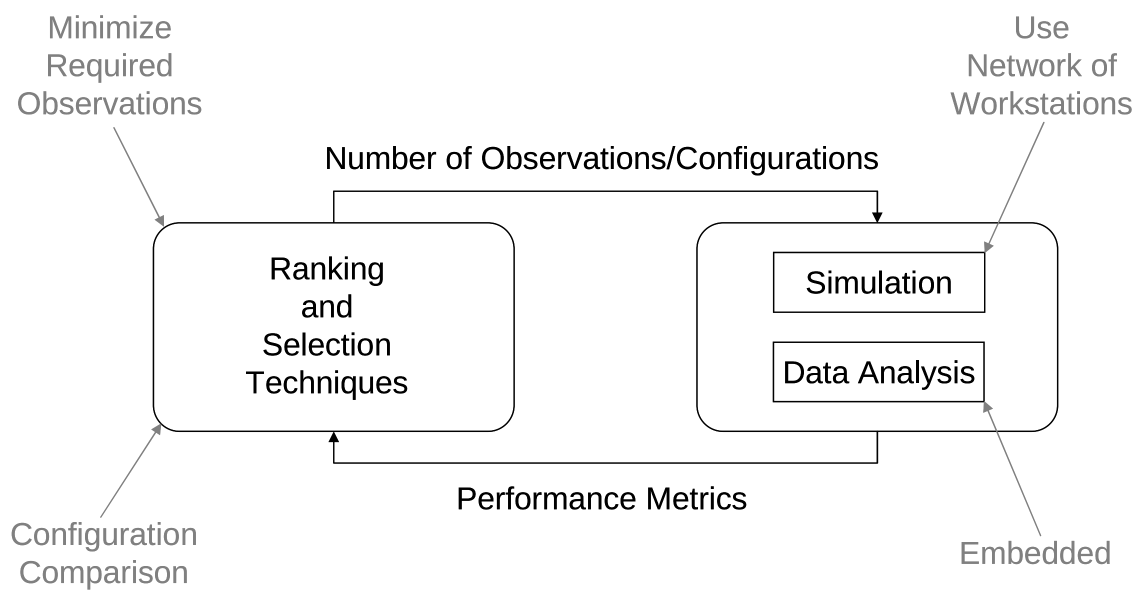
STAR – Standard Terminal Arrival Route

System – a regularly interacting or interdependent group of items forming a unified whole – real-world process of interest

SUMMARY

This thesis describes adaptive simulation control techniques that differentiate between competing system configurations. Here, a system is a real-world environment under analysis. In this context, proposed modifications to a system denoted by different configurations are evaluated using large-scale hybrid simulation. Adaptive control techniques, using ranking and selection methods, compare the relative worth of competing configurations and use these comparisons to control the number of required simulation observations. Adaptive techniques necessitate embedded statistical computations suitable for the variety of data found in detailed simulations, including hybrid and agent-based simulations. These embedded statistical computations apply efficient sampling methods to collect data from simulations running on a network of workstations. The National Airspace System provides a test case for the application of these techniques to the analysis and design of complex systems, implemented here in the Reconfigurable Flight Simulator, a large-scale hybrid simulation. Implications of these techniques for the use of simulation as a design activity are also presented.

The figure below is a graphical summary of this method. Given a properly modeled configuration of some real-world system, simulation of the model provides predictive insight into actual system performance. Embedding data analysis within the simulation facilitates runtime analysis of practitioner defined metrics. Runtime knowledge of these metrics enables adaptive control techniques, such as ranking and selection, to minimize the number of observations required to compare competing configurations. Using a network of workstations with parallel and distributed simulation methods makes best use of available computational capacity for a given experiment.


Minimize Required Observations
Use Network of Workstations
Number of Observations/Configurations
Ranking and Selection Techniques
Simulation
Data Analysis
Performance Metrics
Configuration Comparison
Embedded

Adaptive Control of Large-Scale Simulations

This method can be generalized to any endeavor comparing two or more model configurations, using simulation, during analysis and design. The key contribution of this research is the integration of academic fields to improve complex system analysis. Additionally, ranking and selection methods are extended. Lastly, experimental sampling methods are developed suitable for agent-based simulation timing mechanisms.

CHAPTER 1

INTRODUCTION

Increasing use of simulation for both design and analysis motivates models capable of increasingly realistic representations of complex systems. For example, one method for obtaining increased realism is the use of hybrid simulations. Modeled system performance can be inferred from simulation output in a variety of manners from simple queuing times to multifaceted compliance with regulations or constraints.

Hybrid simulations, that is, simulations capable of simultaneously including discrete-event and continuous-time models, allow for cost-effective and detailed analysis of systems that involve complex interactions between heterogeneous entities. Agent-based modeling is one method for describing such heterogeneous entities. In this paradigm, each individual agent autonomously pursues a goal and also interacts with other agents inside the simulation. Agent-based modeling provides an inherently modular method for high-fidelity simulation of complex systems. This approach, however, requires the inclusion of a range of models with varied output data types such as discrete and continuous. For example, in an air traffic simulation an appropriate discrete state variable may be the number of aircraft arrivals into a defined airspace, while a continuous variable of interest might be the minimum separation between two aircraft.

Detailed hybrid simulations, including agent-based simulations, require an increase in both size and runtime. Frequently, the amount of simulation output is determined by the availability of computational capacity. Subsequent data analysis, commonly done as a separate activity, often reveals either insufficient or excess observations for the required statistical comparison. Therefore, embedding statistical estimators within a simulation can ensure computationally efficient sampling without requiring storage and post hoc analysis.

Incorporating an adaptive control technique, such as a Ranking and Selection (RS) method, offers an additional avenue for increased computational efficiency. RS methods calculate the number of required observations, thus ensuring that statistically sound comparisons are made with modest computational expense. The methods presented here are sequential and appropriate for general stationary output processes. A new adaptive control technique is developed here that relies on embedded statistical estimators to calculate the number of required observations for each simulated configuration. Additionally, the control technique differentiates in an adaptive manner between competing configurations by identifying which configurations do not warrant further analysis, potentially saving computational resources.

Bringing together hybrid simulation models, embedded statistical analysis, and adaptive control techniques improves the application of simulation to the analysis and design of complex systems. This improvement is realized in terms of computational reduction and statistically valid comparison of competing configurations. Additionally, this method creates an environment conducive to Parallel and Distributed Simulation (PDS), although the control techniques employed here do not require the strict time management generally required in PDS. Instead, experiments can be implemented on a Network of Workstations (NOW) that coordinates observation sampling from complementary simulations.

An existing hybrid simulation model of the National Airspace System (NAS) provides a relevant test case (Pritchett and Ippolito, 2000). Specifically, as part of this research, the Reconfigurable Flight Simulator (RFS) has been extended for embedded statistical computations and adaptive control techniques. The analysis of arrival routing configurations for Atlanta International Airport (ATL) is presented as a demonstration.

Figure 1 provides a graphical summary of this method. Given a properly modeled configuration of some real-world system, simulation of the model provides predictive insight to actual system performance. Embedding data analysis within the simulation facilitates runtime analysis of practitioner defined metrics. Runtime knowledge of these metrics enables adaptive analysis and control techniques, such as ranking and selection, to minimize the number of observations required to compare competing configurations. Using a network of workstations in a parallel and distributed simulation manner makes the best use of available computational capacity for a given experiment.

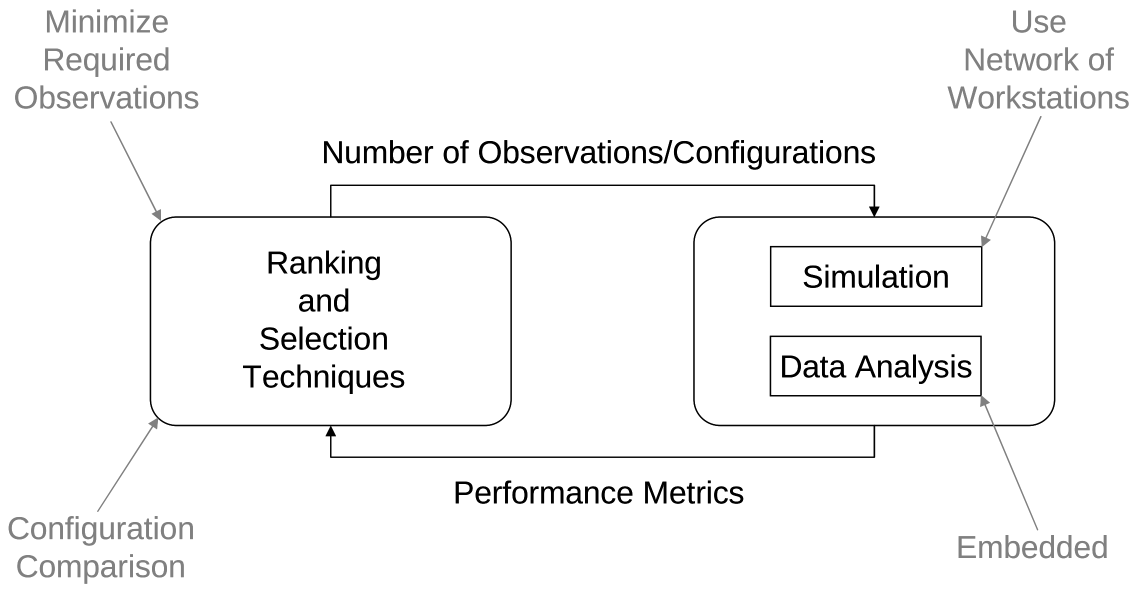

Minimize Required Observations
Use Network of Workstations
Number of Observations/Configurations
Ranking and Selection Techniques
Simulation
Data Analysis
Performance Metrics
Configuration Comparison
Embedded

Figure 1: Adaptive Control of Large-Scale Simulations

This comparative method may be applied to modeled complex systems differentiated by a unique performance metric. For example, continuous-valued profit measures or the relative physical separation of modeled components could identify desired performance for a particular system. The comparative method can also incorporate derived metrics from discrete variables, such as average throughput. This method relies on observations of estimated mean values which must exhibit characteristics enabling the application of ranking and selection methods. Specifically, ranking and selection method performance must be robust for the achieved normality and serial correlation of these observations. Lastly, obtaining observations must be computationally tractable.

1.1 Research Objectives

The objectives of this research can be summarized as follows:

- Development of efficient and accurate embedded statistical estimators that enable fully sequential adaptive control techniques providing comparative analysis between competing simulated model configurations.
- Extension of adaptive control techniques, in this case ranking and selection methods, that differentiate between competing simulated model configurations and also calculate the number of required observations.
- Development of a distributed simulation architecture ensuring the best use of computational capacity.
- Integration of these components, thereby extending the use of simulation as an analysis and design tool for complex systems.

1.2 Research Outline

The remainder of this dissertation is organized as follows. Chapter 2 presents a literature review of simulation as a design process, adaptive analysis and control techniques, embedded statistical encapsulation, and a discussion on parallel and distributed simulation. Chapter 3 details the developed distributed simulation architecture. Chapter 4 highlights extension of ranking and selection methods as a control technique. Chapter 5 presents the application of the developed methods to an existing large-scale simulation of the National Airspace System. Chapter 6 summarizes the effort and discusses future research directions.

CHAPTER 2

BACKGROUND

Analysis of complex systems requires the integration of several academic fields for their efficient and accurate study. Hybrid simulation, comprised of discrete and continuous valued variables, offers a cost-efficient method for complex system analysis. Timing mechanisms for agent-based simulations, synchronous or asynchronous, impact the appropriate sampling of experimental observations. Embedded statistical sampling methods need to be efficient in terms of memory storage requirements as well as accurate in terms of estimation. Adaptive control techniques, such as Ranking and Selection (RS), allow for efficient computation usage by terminating analysis of system configurations that are no longer competitive. Furthermore, relatively “fast” computational results may be obtained from a Network of Workstations (NOW) implementation that builds on current Parallel and Distributed Simulation (PDS) techniques. This chapter focuses on the discussion of these perspectives and their relation to the increasing use of simulation in the design of complex systems.

2.1 Simulation

Law and Kelton (2000) define simulation as a technique using computers to imitate various kinds of real-world facilities or processes. Generally, simulation offers a relatively inexpensive and fast technique for gaining insight into the performance of complex systems. In this context, a system represents a process or facility under scrutiny and the simulated model of this system is composed of logical and mathematical constructs that define system functionality. Unless an analytic/numerical solution is possible or the system may be observed while in operation, simulation is required to estimate system performance.

Simulation as an analytic tool is ubiquitous. However, the actual implementation of this tool spans a variety of methods. Separate discrete and continuous modeling techniques are already well established and have also been combined in hybrid simulations to create increasingly complex models (Sanchez and Lucas, 2002). Use of the agent-based paradigm further allows the practitioner to more accurately model many types of systems and identify emergent behavior. This section briefly discusses these methods as they relate to the modeling of complex systems.

2.1.1 Common Types of Simulation

Discrete-event simulation models a system as having state variables that change by discrete amounts at discrete points in time. For example, an appropriate state variable for discrete modeling might be the number of customers in a queue. Performance metrics of a discrete model include delay, number of waiting entities, system throughput, and resource utilization (Fishman, 2001). Discrete-event simulations are widely used in many domains including manufacturing systems, military operations, and transportation networks. This type of simulation is well suited for modeling the stochastic nature of arrivals, waiting/servicing, and departures in a system.

Continuous-time simulation models a system where continuously valued state variables change over time. Continuous-time simulation is effective for modeling dynamic behavior such as vehicle movement. For example, modeling aircraft flight dynamics includes state variables describing characteristics such as attitude and heading. Differential equations are used to update variables associated with dynamic behavior.

Several numerical integration routines, such as Runge-Kutta, can be used to update these state variables during the simulation (Chen, 2000).

Hybrid simulation models include both discrete and continuous state variables. This allows for a more realistic representation of a system by modeling both discretely and continuously varying state variables. For example, a hybrid model uses differential equations for the internal dynamics of each vehicle and discrete state variables to count the number of vehicles at a particular location. However, the increased complexity of hybrid models entails longer development and execution time.

Agent-based simulation is one form of simulation that models a system through the use of agents, often with hybrid models. An agent in this context is an autonomous entity that interacts with other agents and the environment in the pursuit of a goal or set of goals. For example, an aircraft agent modifies its flight to avoid other aircraft and terrain during its approach to a given airport. Agent-based simulations are increasingly being applied to model a variety of systems, including telecommunications, business processes, control of mobile robots, and military operations (Logan and Theodoropoulos, 2001).

Pritchett et al. (2002) discuss an agent-based simulation of the National Airspace System (NAS) comprised of heterogeneous entities. The NAS is an example of a complex socio-technical system composed of various entities such as pilots, controllers, technical devices, and aircraft. Here, a socio-technical system denotes one that contains both human and machine components. This example highlights the complexity of agent-based simulations and the motivation to make the best use of models that require both extensive developmental effort and relatively “long” execution time.

2.1.2 Simulation as a Design Activity

Simulation is often used iteratively during the design process. This section highlights the general use of simulation in electronic circuit analysis, manufacturing, aircraft operations, and military endeavors. Each application of simulation facilitates product and/or process improvement.

One example of hybrid simulation practice is electronic circuit analysis. The goal of circuit simulation is pre-manufacturing verification of potential performance (Pillage, et al., 1995). Connectivity within the circuit is modeled by logical gates that have discrete behavior. Timing mechanisms for these logical gates are generally event driven. For instance, a gate is opened at a particular time. Circuit elements, such as resistors and capacitors, are evaluated by continuous parameters such as voltage or current updated by numerical integration. Typical use of circuit simulation entails sensitivity analysis; one varies a particular resistor parameter and observes how the current changes in the circuit. This capability has enabled modern Very Large-Scale Integrated (VLSI) circuit design.

The use of simulation in manufacturing processes has grown dramatically in recent years. Increased demand for high quality goods on short notice is one factor motivating the use of simulation. In this realm, simulation is often used during product acquisition to support:

- Requirement definition and analysis
- System engineering
- System development process
- System testing
- System training

To that end, these simulations necessitate a hybrid capability that is interoperable between agencies participating in the acquisition process (NRC, 2002).

The use of simulation in the design of aircraft operations can be beneficial in multiple areas. Allocation of ground delay to mitigate costly airborne congestion is one example (Kleinman, Hill, and Ilenda, 1998). In this example, various methods of allocating ground delay were assessed by the use of simulation and an optimization algorithm. Evaluation of recovery policies following unforeseen traffic disruptions is another example. SimAir is a discrete-event simulation that models daily airline operations to include aircraft, crews, passengers, and disruptions (Schaefer, et al., 2002). Simulation also serves as a stimulus for development of aircraft-related technologies such as hydraulics and for real-time human-in-the-loop simulators (Rolfe and Staples, 1986). Note these simulations are often hybrid in nature with continuous variables such as position and velocity accompanied with discrete variables such as the number of aircraft arrivals.

Military simulations generally focus on combat operations. One focus is on attrition rates incurred during a force-on-force confrontation. Extension of two-sided to *n*-sided conflict analysis is of recent national interest (Brandt and Roland, 1993). Another focal point for military simulations is weapon effectiveness. Here, probabilistic analysis of potential detection or successful engagement during a confrontation is combined with cost analysis during the acquisition process. Lastly, large-scale combat simulations facilitate military staff operational training.

2.1.3 Simulation Summary

Large-scale hybrid simulations that mimic complex systems offer an efficient method for design and analysis. The previous section highlights the flexibility of simulation as an analysis tool suitable for a wide variety of domains. Note the appropriate use of simulation relies on model parameterization and correct output analysis. However, the relevance of simulation relies on the computational tractability and actual development costs. There are pitfalls in the current use of hybrid simulation. First, embedded statistical controls are infrequently implemented. Rather, analysis is usually done post hoc and often by separate agencies. Also, the sheer complexity of these simulations entails the use of vast computational resources.

2.2 Sampling from Hybrid Simulations

Hybrid simulations provide a generalized approach to modeling real-world systems as they allow for both discrete and continuous state variables. Discrete events, such as vehicle arrivals, often involve metrics derived from queue size. On the other hand, continuous state variables, for example vehicle velocity, involve metrics based on maximum, minimum, or average values. Entities in hybrid simulations commonly have differentiable update rates depending upon both their internal dynamics and interactions with other entities.

Likewise, the autonomous aspect of agent-based simulation has inspired novel timing mechanisms (Lee, Pritchett, and Goldsman, 2001). In addition to involving hybrid dynamics, agent-based simulations may use either asynchronous or synchronous timing mechanisms. Both types of mechanisms necessitate sophisticated statistical sampling methods as detailed in the following sections.

2.2.1 Synchronous Sampling

Synchronous sampling entails obtaining observations at predetermined time steps that are periodic. The following example assumes post hoc sampling where all observations are stored for later analysis. In Figure 2 below, objects 1 through 4 are updated at 0.2 second increments. Sampling occurs every 0.5 seconds with an overlap of 0.2 seconds. For instance, a sample taken at 9.5 seconds includes observations with update times from 9.4 to 9.6 seconds. Note that at a simulation time of 9.5 seconds there are eight observations while there are only four at the 10.0 second sampling. This difference has minimal, if any, effect on the measure of any state variable. However, calculations must be based on the number of observations rather than the number of objects. Also, the practitioner must ensure the sampling rate and overlap will indeed capture observations.

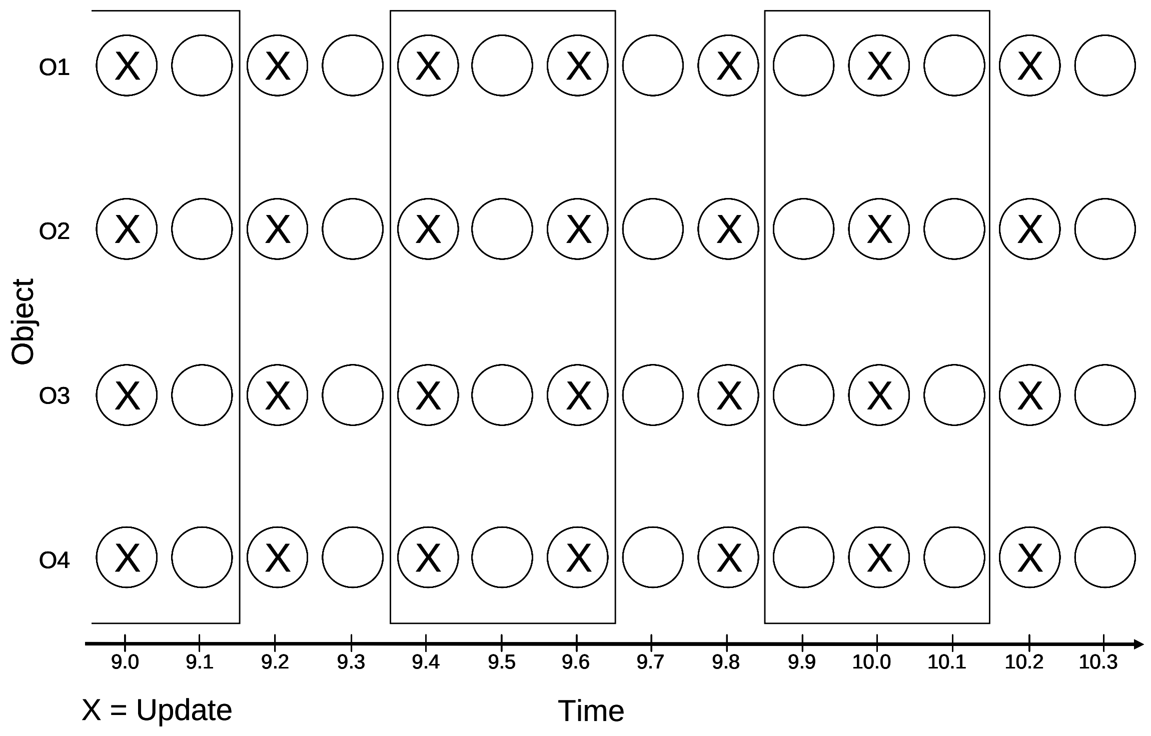


Figure 2: Synchronous Sampling

2.2.2 Asynchronous Sampling

In general, synchronous timing methods for large-scale simulations are computationally inefficient because all agents update at specified time steps regardless of the necessity. Asynchronous timing methods can also provide correct model and simulation results if update times are managed by agents in accordance to their internal dynamics and interactions. For example, Figure 3 shows a possible 2-dimensional spatial separation scenario with six vehicles. Obviously, vehicles 1 and 2 must be checked more often than vehicles 5 and 6 to measure if safe separation has been lost.

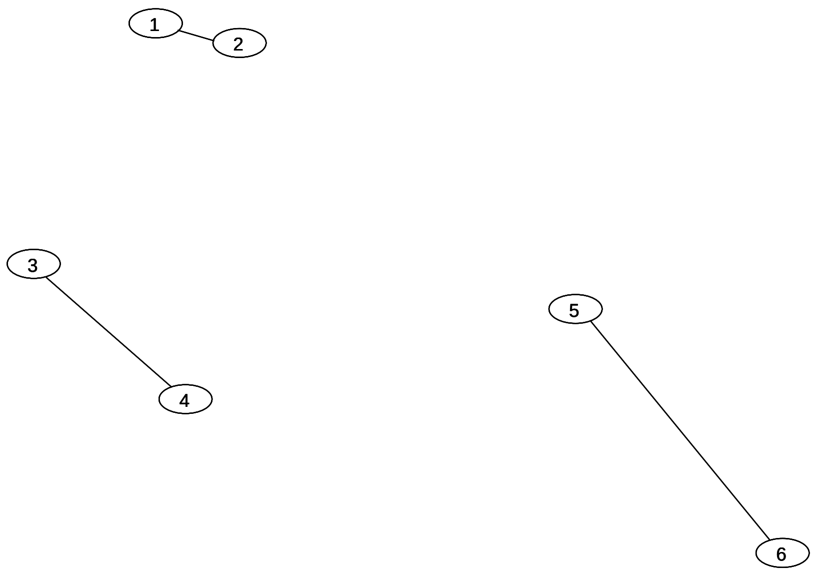


Figure 3: Spatial Separation Asynchronous Update Example

Asynchronous statistical sampling creates a more perplexing problem. Figure 4 below details a possible sampling scenario. Objects 1, 2, and 3 update at periodic intervals, but their update rates are different. Hence, sampling at set intervals could bias the estimated mathematical distribution to more frequently updated objects. Likewise, object 4 in this example is updating at intervals that are not constant. Objects of this nature add randomness to the results obtained via a constant sampling rate.

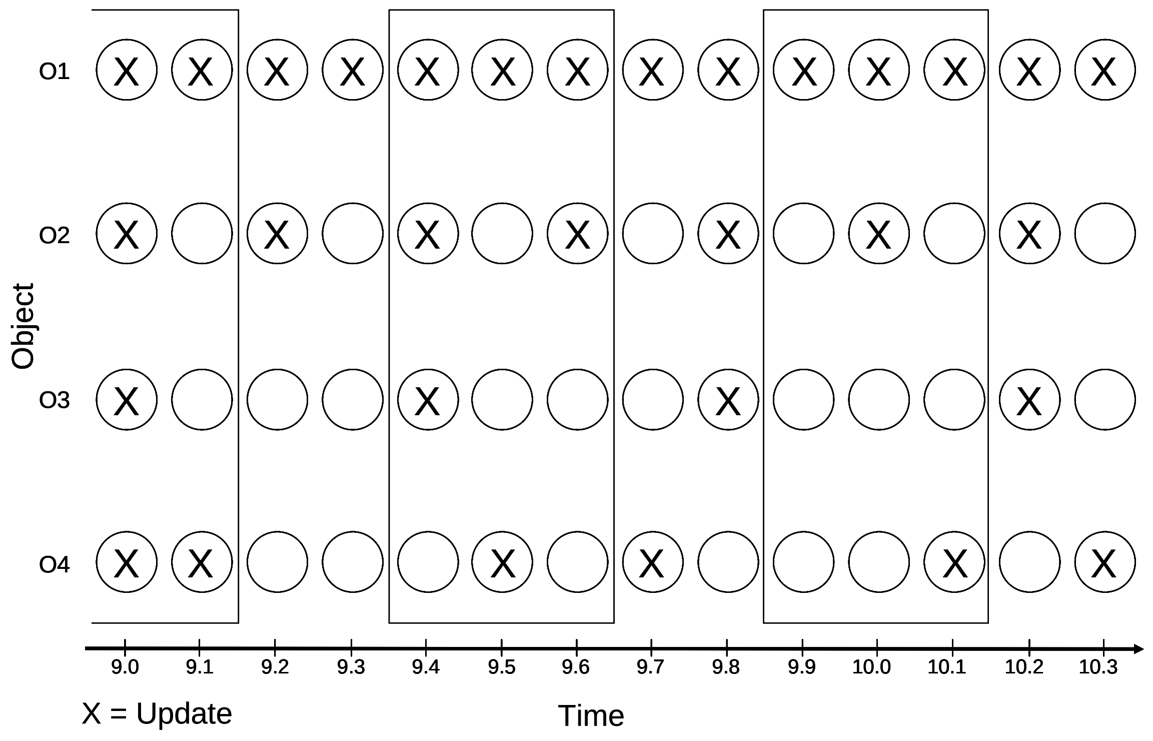


Figure 4: Asynchronous Sampling

2.2.3 Combined Sampling

Sequential combination of simulation output from these sampling methods entails several assumptions. First, the simulated configurations must be of identical models. Second, generated random numbers from the simulations must not significantly overlap or the obtained observations may be redundant. Note that combining variance estimators, discussed later, generally results in higher estimated values due to the decreased degrees of freedom. However, underestimation of variance may also occur when combining simulation output.

Combining simulation output to predict long-term systemic performance also relies on assumptions about the complex system. For example, the analysis of arrivals into a facility can be modeled by seasonal factors such as the hours of daylight truncated by day increments. To some extent, day-to-day increments of this simulated output may be combined. Yet, missed systemic failures, with subsequent recovery, are the major issue with this assumption. For example, a severe weather incident may stop all arrivals on a particular day, and recovery on the next day will entail increased traffic flow. Combining simulation output in the discussed manner would fail to capture such effects. However, combining simulation output potentially speeds complex system analysis.

2.2.4 Sampling Summary

Simulation timing mechanisms complicate observation sampling from an experiment. Sampling methods should be generalized to handle any timing mechanism and type of variable. Additionally, simulation-specific diagnostic tests are required to select appropriate sampling rates and the overlap size. In this context, an appropriate rate and overlap ensure accurate estimation of the underlying process. Therefore, development of relatively simple sampling methods that avoid storage of historical data and preclude over/under flow of variables are needed; these methods should also be computationally efficient, accurate, and robust.

2.3 Parallel and Distributed Simulation

Parallel and Distributed Simulation (PDS) has been studied for many years in an effort to speed increasingly complex simulation models. PDS research has primarily focused on manipulating sequential simulations and unifying coupled simulation processes for discrete-event systems. Manipulation of sequential simulations is generally accepted for queuing systems or for the mass replication of a particular simulation configuration. Combining separate but interacting simulations spans sophisticated synchronization algorithms and prescribed interfaces such as the DOD High Level Architecture (HLA). Fujimoto (2000) provides a comprehensive discussion of current PDS techniques that is briefly highlighted in this section.

A key issue in most PDS techniques is the synchronization of a set of logical processes (LP) running on separate processors. Note that LP assignment to participating processors is generally done by temporal or spatial decomposition. Temporal methods logically separate distinguishable simulation scenarios by time, such as decomposition of aircraft arrivals by day increments at a major airport. The issue with decomposition by this method relies on tying the terminating conditions of one temporal LP to the initial conditions of another. Spatial decomposition, on the other hand, divides complex simulations into geographically separate LPs, such as separation of arrival patterns into a given airport, in an effort to distribute the computational load. This method of decomposition requires spatially adjoining simulations to communicate or pass entities between LPs. In general, between-LP interaction is required at the boundaries created by the decomposition method.

The goal of synchronization in this context is to avoid causality errors from out-of-order event processing. For example, under temporal decomposition, if one LP computed an event that impacted a previous event on a different LP, then a causality error might exist. The two classes of algorithms that address this issue are called conservative and optimistic. Conservative synchronization algorithms strictly enforce the event processing in the associated LPs to avoid causality errors. On the other hand, optimistic algorithms detect causality errors at runtime with a subsequent roll back mechanism to "undo" the error; Time Warp is a well-known example (Fujimoto, 2000). Note that these algorithms may entail significant computational expense to accurately roll back the LP. Regardless of the method, synchronization methods rely on strict interpretations of time in the participating LPs, or federate in HLA vernacular, for successful and repeatable execution.

Other PDS implementations focus on compiler or operating system kernel modifications to speed LPs. For example, Carothers (2002) implemented a PDS called Extreme Simulation (XSim) on the Linux operating system. XSim is promising in terms of simplified kernel modification and virtual memory management, but is limited by the cost of redeveloping existing simulation models to this paradigm.

Another possible PDS architecture allows for heterogeneous processor contributions to a given experiment. Specifically, contributing processors simulate differentiable model configurations of a complex system. Unlike temporal and spatial decomposition PDS methods that contain coupled dynamics, this approach incorporates independent execution of the simulations. The lack of coupled dynamics with this approach avoids causal synchronization issues. Beyond mere mass replication of a particular simulation, this method naturally acquires computational capacity for configuration comparison. Computational load sharing in this manner is similar to previous efforts by Karatza and Hilzer (2002).

2.4 Metric Selection

Metrics define measurable criteria for organizational analysis. Obviously comparative analysis between competing configurations requires selection of a metric valued by the practitioner. These metrics can be based on a mix of discrete and continuous valued variables. Analysis methods must accommodate these varying metrics. For example, the following are a subset, and "non-exhaustive" survey, of potential metrics for analysis of the National Airspace System (NAS):

- Safety
  - Recordable injuries/fatalities
  - Lost workday cases
  - Aircraft damage
  - Accidents/incidents
  - Aircraft spacing
  - Go around frequency
- Schedule performance
  - Aircraft on time
  - Arrival/departure delay rate
  - Aircraft turn time
- Cost/Benefit
  - Passenger revenue from tickets
  - Inconvenienced passenger costs
  - Fuel consumption

Clearly, there are numerous metrics available to the analyst of air transportation, and each may be analyzed in a number of ways. For example, the analyst may require a measurement of a minimum, maximum, average, and/or count of a variable.

The goal of this research is not to extend metric development. Instead, the demand for analysis of a variety of metrics motivates two aspects of this research. First, analysis of complex systems requires metrics encompassing discrete and continuous-valued variables. Second, the use of runtime-determined metrics encourages modularity and reuse.

2.5 Embedded Statistical Analysis

Adaptive control techniques often require calculations on both individual and batched observation data. To that end, a relatively simple data encapsulation method is discussed later in this thesis that does not require use of historical experimental observation values, but instead maintains only current state variables and certain summed values. This embedded method allows for both estimator calculations and availability of these estimators at each simulation time step along with inherent reduction of memory usage.

Embedded statistical estimators enable the acquisition of batched observations. These batched observations, under certain conditions, exhibit characteristics assumed for the appropriate application of adaptive control techniques. Specifically, data batching methods facilitate the acquisition of normally distributed batched observations. Also, an appropriate sampling rate allows us to obtain unbatched observations correlated at a manageable level. Note that variance estimators for correlated data are typically biased. Simulation-specific diagnostics determining these values are discussed later in this thesis. The remainder of this section details methods for the acquisition of normally distributed simulation output that can also avoid autocorrelation issues.

Embedding this data acquisition method within a steady-state simulation whose output is neither independent nor identically distributed requires assumptions on simulation initialization. Law and Kelton (2000) suggest truncating early data in a simulation as one method of avoiding initialization bias. Embedded statistical estimators are used to compare *k* competing simulated configurations where $i = 1,2,\ldots,k$. Given $X_{i1}, X_{i2},\ldots$ as the simulation output from a single replication of the *i*th alternative, then after appropriate initialization the following assumptions hold:

Stationarity: $X_{i1}, X_{i2},\ldots$ forms a stationary stochastic process.

(Strong) Consistency: $\overline{X}_i(r) \to \mu_i$ as $r \to \infty$ with probability one, where μ_i is the steady-state mean from system *i* and $\overline{X}_i(r)$ is the sample mean based on *r* observations from system *i*.

Functional Central Limit Theorem (FCLT): There exist constants μ_i and $v_i^2 > 0$ such that

$$\frac{\sum_{j=1}^{\lfloor rt \rfloor}\left(X_{ij} - \mu_i\right)}{\sqrt{r}} \Rightarrow v_i W(t) \tag{1}$$

for $0 \le t \le 1$, where $\Rightarrow$ denotes weak convergence and *W*(t) is a standard Brownian motion (Weiner) process (Glynn and Iglehart, 1990).

For this effort, comparisons are made on steady-state means $\mu_1, \mu_2, \ldots, \mu_k$, which is reasonable due to the consistency assumption. The variance parameter, v_i^2, can be estimated by batch means, overlapping batch means, and standardized time series methods. Note that variance estimation from a single long simulation run ameliorates somewhat the issue of initialization bias. The following techniques provide estimators for the asymptotic variance constant $v_i^2 \equiv \lim_{r\to\infty} rVAR\left(\overline{X}_i(r)\right)$.

2.5.1 Batch Means

If *n* observations $X_{i1}, X_{i2}, \ldots, X_{in}$ are divided into *b* batches of length *m*, then the *j*th batch mean from system *i* is:

$$\overline{X}_{i,j,m} \equiv \frac{1}{m}\sum_{p=1}^{m} X_{i,(j-1)m+p} \tag{2}$$

The observations $X_{i,(j-1)m+1}, X_{i,(j-1)m+2}, \ldots, X_{i,m}$ comprise the *j*th batch, $j = 1,2,\ldots,b$, for system *i*. For $b > 1$, the batch means variance estimator is:

$$mV_B^2 \equiv \frac{m}{b-1}\sum_{j=1}^{b}\left(\overline{X}_{i,j,m} - \overline{X}_i(n)\right)^2 \xrightarrow{D} \frac{v_i^2 \chi_{b-1}^2}{b-1} \tag{3}$$

where χ_d^2 is a chi-squared random variable with $d = b-1$ degrees of freedom and $\xrightarrow{D}$ indicates convergence in distribution as *m* becomes large (Glynn and Whitt, 1991).

2.5.2 Overlapping Batch Means

Consider all batch means of the form:

$$\overline{X}_i(j,m) \equiv \frac{1}{m}\sum_{p=0}^{m-1} X_{i,j+p} \tag{4}$$

The observations $X_{i,j+1}, X_{i,j+2}, \ldots, X_{i,j+m-1}$ comprise the *j*th overlapping batch for $i = 1,2,\ldots,k$ and $j = 1,2,\ldots,n-m+1$ for system *i*. The overlapping batch means variance estimator is:

$$mV_O^2 \equiv \frac{nm}{(n-m+1)(n-m)}\sum_{j=1}^{n-m+1}\left(\overline{X}_i(j,m) - \overline{X}_i(n)\right)^2 \tag{5}$$

Note that $mV_O^2 \approx \frac{v_i^2 \chi_d^2}{d}$ where χ_d^2 is a chi-squared random variable with $d = \lfloor 3(b-1)/2 \rfloor$ degrees of freedom (Meketon and Schmeiser, 1984).

2.5.3 Standardized Time Series

For $i = 1,2,\dots,k$, $j = 1,2,\dots,b$, and $h = 1,2,\dots,m$ the hth cumulative mean from batch *j* of system *i* is:

$$\overline{X}_{i,j,h} \equiv \frac{1}{h}\sum_{p=1}^{h} X_{i,(j-1)m+p} \tag{6}$$

For $i = 1,2,\dots,k$, $j = 1,2,\dots,b$, and $0 \le t \le 1$ the standardized times series from batch *j* of system *i* is:

$$T_{i,j,m}(t) \equiv \frac{\lfloor mt \rfloor \left(\overline{X}_{i,j,m} - \overline{X}_{i,j,\lfloor mt \rfloor}\right)}{v_i \sqrt{m}} \tag{7}$$

The weighted area under the standardized time series formed by the *j*th batch of observations from system *i* is:

$$A_{i,j}(t) \equiv \frac{v_i}{m}\sum_{l=1}^{m} w(l/m) T_{i,j,m}(l/m) \tag{8}$$

where $w(t) \equiv \sqrt{840}\left(3t^2 - 3t + 0.5\right)$, obtained from Goldsman et al. (2002), is a wise choice for a weighting function to reduce bias. The weighted area variance estimator is:

$$mV_A^2 \equiv \frac{1}{b}\sum_{j=1}^{b} A_{i,j}^2 \xrightarrow{D} \frac{v_i^2 \chi_b^2}{b} \tag{9}$$

2.5.4 Embedded Statistical Analysis Summary

The embedded methods presented in this section enable encapsulation of defined performance metrics. Furthermore, these methods provide normally distributed observations under certain conditions. Generally, a sufficiently large batch size is required for the batch means method. This guarantees normally distributed batched observations by the Central Limit Theorem. The overlapping batch means method requires sufficiently large batch size along with a large ratio of raw sample size to batch size. The ratio requirement mitigates inherent convergence issues with this method. Note the application of these methods to a new simulation necessitates diagnostics to verify normality and variance convergence before use with adaptive control techniques.

2.6 Adaptive Control Techniques

The goal of any selection, screening, and multiple comparison problem is to determine the "best" of several competing configurations. In this context, a configuration implies that we have two or more competing systems that are compared by the mean value of some metric describing performance, where simulation is required to assess the value of this metric. Bechhofer et al. (1995) highlight several problem formulations appropriate to various experimental designs. Here, focus is on the indifference-zone formulation where the objective is to select the configuration with the highest/lowest (interpreted "best") expected value. In this realm, an expectation offers insight on long-term performance.

The practitioner provides (δ^*, P^*), where δ^* is the indifference-zone parameter and P^* denotes the threshold desired probability of correctly identifying a difference between configurations. Note that the indifference-zone indicates some comparative region where the practitioner would not discriminate between configurations. Also, the threshold probability, P^*, can be interpreted as a $1-\alpha$ confidence interval when configuration mean values do in fact differ by at least δ^*.

Ranking and Selection (RS) methods enable adaptive control of this multiple comparison problem. Ultimately, RS methods determine the number of required observations necessary for statistically rigorous comparison of competing simulated configurations. RS methods may be single or multi-stage. In this context, a stage denotes the execution of a simulated configuration for a number of observations. A single-stage method determines the number of required observations from parameters determined by the experimenter. Adaptive control is not possible with a single-stage RS method. However, a multi-stage RS method updates the required number of observations from simulated configuration output, thereby enabling adaptive control of the comparison problem.

To highlight a single-stage RS method, if we assume random data from a normal distribution with known variance, σ^2, then the classical Bechhofer (1954) method is appropriate. This method determines the number of required observations, *n*, from the following:

$$n = \left\lceil 2\left(\frac{\sigma Z_{k-1,0.5}^{(1-P^*)}}{\delta^*}\right)^2 \right\rceil \tag{10}$$

$Z_{k-1,\rho}^{(1-P^*)}$ is the $(1-P^*)$ equicoordinate point of the $k-1$ dimensional multivariate standard normal distribution with off-diagonal correlation ρ. Values for $Z_{k-1,\rho}^{(1-P^*)}$ may be obtained from table lookup or from the application presented in Appendix B. Here, *k* is the number of configurations in contention at the start of the experiment. For example, if the variance, σ^2, is known to be 2.25 and the experimenter sets $\left(\delta^*, P^*\right) = \left(0.30619, 0.95\right)$ with $k = 6$ configurations, then 262 observations are estimated for statistical comparison. Examples of two-stage and multi-stage methods follow.

If the variance of a predetermined metric is unknown, then Rinott's method (1978) provides a well known two-stage technique for comparing configurations. This method relies on the assumptions that obtained data are independent, identically distributed, and from a normal distribution. Goldsman et al. (2002) present an extended version of this two-stage method (*R*+) and the extended version of the multi-stage Kim and Nelson (*KN*+) method (2001). Note that batched observations are assumed to be normally distributed for both methods. The following sub-sections detail both methods.

2.6.1 Extended Rinott's Procedure (*R*+)

Setup: Select confidence level $1-\alpha$, indifference-zone parameter $\delta > 0$, first-stage sample size $n_0 \geq 2$, and batch size $m < n_0$.

Initialization: Obtain Rinott's constant (from Bechhofer et al.) $h = h(d, k, 1-\alpha)$, where *d* is the degrees of freedom and *k* is the number of systems.

Obtain n_0 observations X_{ij}, $j = 1,2,\ldots,n_0$, from each system $i = 1,2,\ldots,k$.

For $i = 1,2,\ldots,k$ compute mV_i^2, the sample asymptotic variance of the data from system *i* using estimators discussed in section 2.5.

Let

$$N_i = \max\left\{n_0, \left\lceil \frac{h^2 mV_i^2}{\delta^2} \right\rceil\right\} \tag{11}$$

Stopping Rule: If $n_0 \geq \max_i N_i$ then stop and select the system with the largest estimated mean, $\overline{X}_i(n_0)$, as the best. Otherwise, take $N_i - n_0$ additional observations $X_{i,n_0+1}, X_{i,n_0+2}, \ldots, X_{i,N_i}$ from each system *i* where $N_i > n_0$. Select the configuration with the largest $\overline{X}_i(N_i)$ as the best.

2.6.2 Extended Kim and Nelson's Procedure (*KN*+)

For two systems *i* and *l*, the asymptotic variance of the difference, $v_i^2 + v_l^2$, is calculated by forming the differenced series $D_{ilj} = X_{ij} - X_{lj}$, $j = 1,2\ldots,$ then applying one of the estimators presented in section 2.5 to the series.

Setup: Select confidence level $1-\alpha$, indifference-zone parameter $\delta > 0$, first-stage sample size $n_0 \geq 2$, and batch size $m < n_0$. Calculate

$$\eta = \frac{1}{2}\{[2(1-(1-\alpha)^{1/(k-1)}]^{-2/d} - 1\} \tag{12}$$

Initialization: Let $I = \{1,2,\ldots,k\}$ be the set of systems still in contention, and let $h^2 = 2\eta d$. Obtain n_0 observations X_{ij}, $j = 1,2,\ldots,n_0$, from each system $i = 1,2,\ldots,k$.

For all $i \neq l$ compute mV_{il}^2, the sample asymptotic variance of the difference of systems *i* and *l*.

Let

$$N_i = \left\lfloor \frac{h^2 m V_{il}^2}{\delta^2} \right\rfloor \tag{13}$$

and

$$N_i = \max_{l \neq i} N_{il} \tag{14}$$

Here $N_i + 1$ is the maximum number of observations that can be taken from system *i*. If $n_0 \geq \max_i(N_i + 1)$ then stop and select the system with the largest $\overline{X}_i(n_0)$ as the best. Otherwise, set the observation counter $r = n_0$ and go to Screening.

Screening: Set $I^{old} = I$. Let

$$\begin{aligned} I = \{i : i \in I^{old} \text{ and } \overline{X}_i(r) \geq \overline{X}_l(r) - W_{il}(r), \\ \text{for all } l \in I^{old}, l \neq i\} \end{aligned} \tag{15}$$

where

$$W_{il}(r) = \max\left\{0, \frac{\delta}{2r}\left(\frac{h^2 m V_{il}^2}{\delta^2} - r\right)\right\} \tag{16}$$

Stopping Rule: If $|I| = 1$, then stop and select the system whose index is in *I* as the best. Otherwise, take one additional observation $X_{i,r+1}$ from each system $i \in I$ and set $r = r + 1$. If $r = \max_i N_i + 1$, then stop and select the system whose index is in *I* and has the largest $\overline{X}_i(r)$ as the best. Otherwise, repeat the screening process. Note that variance estimation only depends on data collected in the initialization stage of this method.

2.6.3 Adaptive Control Technique Summary

Adaptive control techniques enable differentiation of competing system configurations. Additionally, these techniques determine the number of required observations necessary to discriminate between competing system configurations. The application of these techniques to a large-scale simulation of a complex system is a relatively new idea. Typically, previous adaptive control techniques of this nature validated performance by the use of simulations mimicking a known process rather than a large-scale simulation modeling a complex system. Limitations to current techniques include reliance on normally distributed data and some type of staged execution. Extension of sequential adaptive control techniques, enabled by embedded estimators, is promising in terms of enhanced computational efficiency.

2.7 Summary

The use of large-scale simulations in the design and analysis of complex systems will be improved by the integration of embedded statistical analysis, adaptive control techniques, and parallel simulation methods. Improvements are in the form of increased computational efficiency and appropriate statistical comparison of competing simulated configurations. Extension of current ranking and selection methods will further increase computational efficiency. Application of this integrated method extends the use of simulation as a design and analysis activity.

CHAPTER 3

DISTRIBUTED COMPUTING METHOD

The Parallel and Distributed Simulation (PDS) techniques discussed in section 2.3 facilitate efficient use of computational capacity. This capacity can be applied to enlarging the scale of a particular simulation or, relevant to this effort, enabling appropriate statistical analysis by the acquisition of sufficient simulated observations. This chapter presents a distributed simulation architecture capable of incorporating Ranking and Selection (RS) methods, discussed in section 2.6, as a control technique enabling efficient analysis of competing simulated configurations. A sample application of this distributed simulation architecture is also presented as a performance demonstration.

3.1 Distributed Simulation Architecture

Ranking and selection methods control the number of observations taken from simulated system configurations to select the "best" system configuration(s) among those in contention. Each simulated system configuration is a separate process that can be distributed to participating processors for execution. Using the indifference-zone formulation, it is possible that several configurations may be selected as the "best" ensuring computational termination. RS methods require synchronization of the statistical estimators only when determining which configurations are still competitive for selection as the "best" and when calculating the number of required observations. This section explores the requirements of RS method implementation in a distributed simulation environment and outlines an architecture meeting these requirements.

3.1.1 Distributed Simulation Implementation

A Network of Workstations (NOW), with each simulating a modeled configuration, can provide suitable computational capacity for a given experiment. Simulation jobs need to be assigned by a central server, or controller. Jobs can be different configurations or the same configuration to allow for simulation replications. The controller coordinates NOW usage by using ranking and selection methods to determine which configurations, and their lengths, to distribute out to participating workstations for execution.

The following distributed simulation architecture allows for heterogeneous workstation contributions. Specifically, contributing workstations simulate different system configurations. Unlike temporal and spatial decomposition PDS methods that contain coupled dynamics, this method only requires independent execution of the simulations. The lack of coupled dynamics within the architecture avoids causal synchronization issues. Beyond mere mass replication of a single configuration, this architecture provides the computational capacity for configuration comparison. Computational load sharing in this manner is related to previous efforts by Karatza and Hilzer (2002).

The implementation of a distributed simulation architecture used here is shown in Figure 5. Each stage denotes the execution of a simulated configuration for a number of observations specified by the RS method. The controller can potentially reschedule logical process execution amongst contributing workstations. Between each stage, the controller compares and discriminates between competing configurations. Of interest in this example is the elimination of one competing configuration from future analysis. For example, in Figure 5, configuration C2 is eliminated from further analysis between stage *k*-1 and stage *k*. The multi-stage aspect of this architecture brings together distributed simulation and RS methods, using embedded statistical analysis in the process.

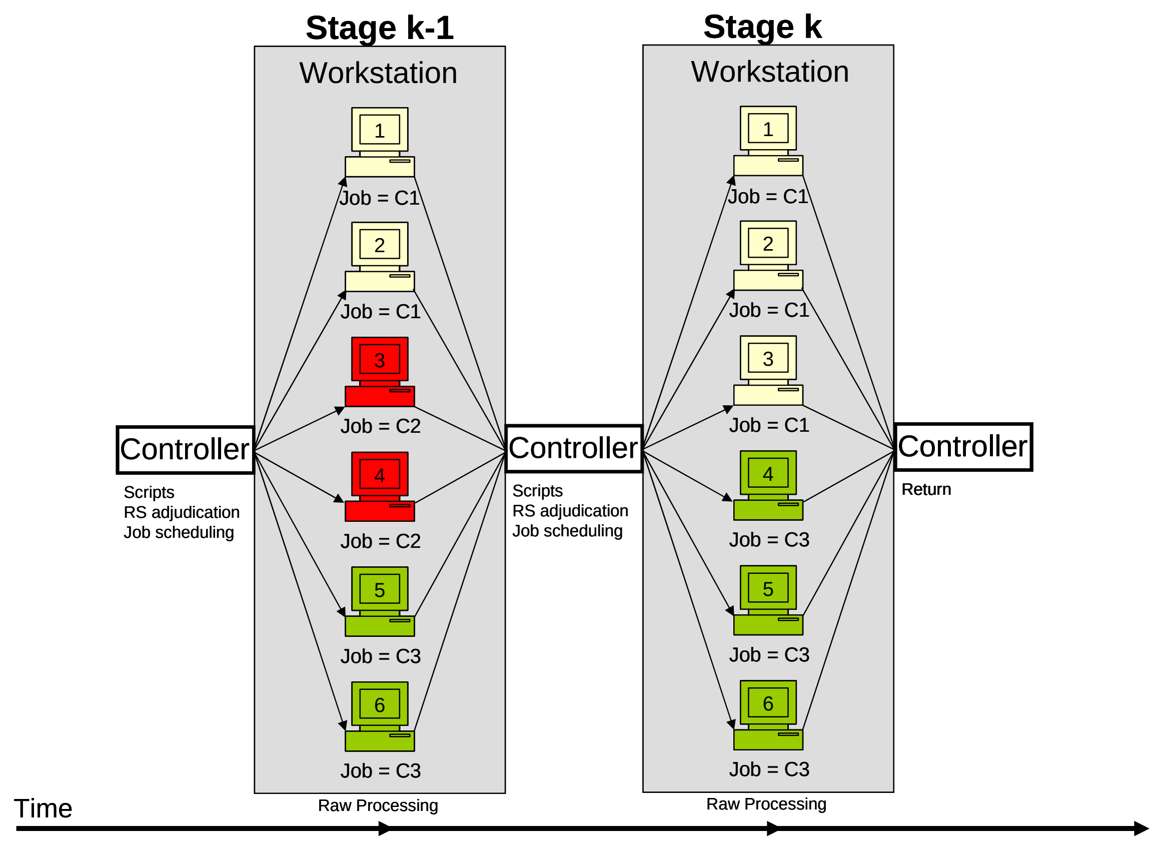


Figure 5: Distributed Simulation Implementation with Ranking and Selection Methods

This implementation requires only the minimal necessary computations for statistically valid configuration comparisons. Distributing simulations in this manner is unique in that it avoids the inherent timing mechanism and synchronization issues faced by most PDS techniques while facilitating "loose" coupling of related processes. Distributed simulation to obtain sufficient data for valid statistical comparison is an extension to the current state of the art.

The specific client-server architecture used here is shown in Figure 6. The server acts as the controller. Competing configurations such as systems A, B, and C are run on participating workstations providing computational contributions to the experiment using simulation executable programs that allow for external control in terms of run, pause, and terminate commands. Additionally, the simulation executable program is required to make embedded statistical calculations.

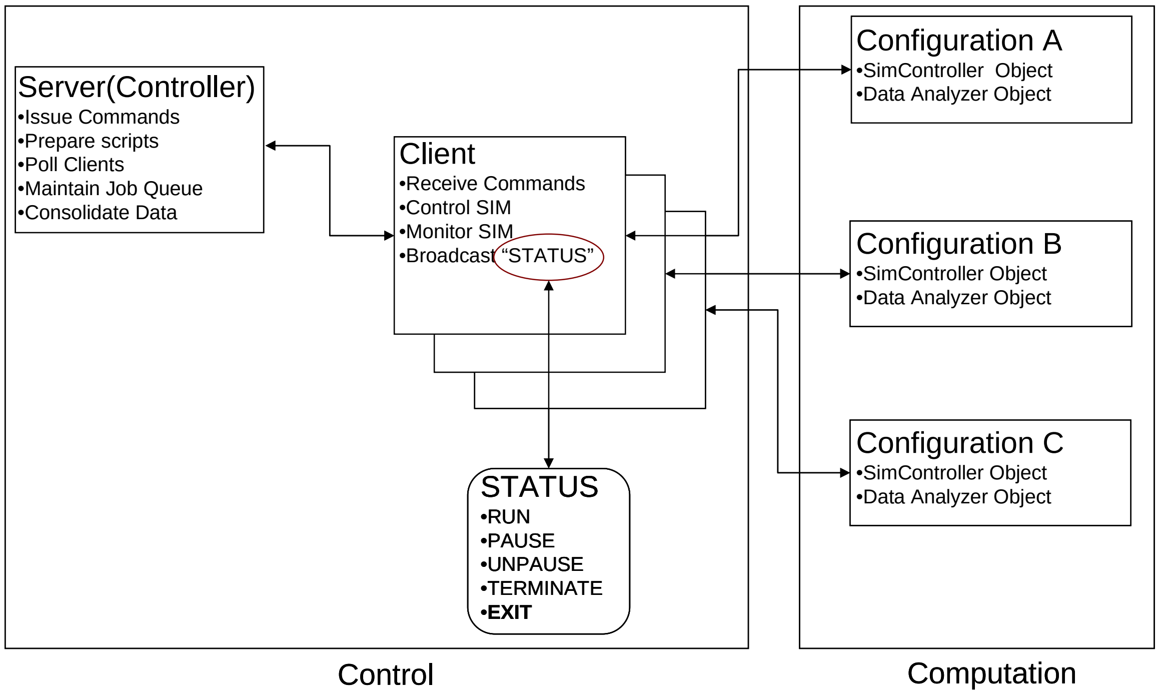


Figure 6: Distributed Simulation Architecture

Participating workstations function as individual clients. Clients act as an interface between the server and the simulation executable program. Each client manages one or more simulation executable program and monitors its associated simulation status. It is assumed that more than one instantiation of the simulation executable program may execute on a participating workstation at any time. For this architecture, only one simulation executable may have a "run" status on each workstation while other simulation executables wait in a “paused” status for client commands. Pausing simulation executables in this manner avoids the transfer of simulation state variables and potential simulation re-initialization. Additionally, the client prepares simulation output for the server. For this implementation, if a client is tasked with more than one job then it must be differentiable in terms of the naming convention. Also, job scheduling on the client is sequential when more than one is assigned.

The server interacts with the clients as it compares different configurations’ metrics in a statistical sense at appropriate intervals. Beyond monitoring status, the server also consolidates simulation output required for ranking and selection methods. Because the sampling intervals can be much greater than time steps within the simulations and because the comparisons are only used to start and end simulation runs, the distributed simulations are much more “loosely” coupled than in most PDS implementations. Therefore, strict time synchronization is not required in this distributed simulation architecture.

The server uses RS methods to calculate the number of required observations and also to discriminate between competing configurations. Here, if one particular configuration is deemed unworthy of further analysis due to poor performance, then it is eliminated from further computational analysis. The server also maintains the status of participating clients. Additionally, the server manages “job” allocation, where a “job” in this case is the simulation sampling requirements for a particular configuration, as detailed in the next section.

Communication between the client and the existing simulation is accomplished through the use of text scripts. Server and clients communicate by the use of an operating system managed TCP/IP Ethernet connection. This generalized approach is extensible. Additionally, it is easily reconfigured for varying experimental designs.

3.1.2 Distributed Simulation Job Queuing

Defining a job as a requirement for a specific number of simulated observations and a machine as a workstation highlights the scheduling problem inherent to this distributed simulation architecture. Typical scheduling problems are NP-hard (Hopp and Spearman, 2000). Assuming simulated configurations are similar, acquisition of first-stage observations requires approximately the same time when using homogeneous processors on the contributing workstations. However, heterogeneous workstation use and later-stage observation requirements obtained from ranking and selection methods complicate the estimation of job duration.

With this distributed simulation architecture, job requirements can be dynamically resized using RS methods. In this context, a job is a quantifiable computational expense, such as running a particular configuration of a simulated model for a specified number of observations. Assuming homogeneous workstations contribute to an experiment, the differing observational requirements, or job size, dramatically increases the difficulty of efficient job queuing. However, the decreased computational expense achieved through the deletion of unnecessary jobs, i.e., simulated system configurations that are no longer competitive, offers increased computational efficiency. In addition, the comparative capability of such a method enables automated design analysis.

This distributed simulation architecture allows for job allocation in several manners. If the practitioner lacks knowledge of simulation computational requirements and believes that combining simulated configuration output is inappropriate, then job allocation is sequential. For example, if there are six configurations and three clients, then client one receives job A, client two job B, etc. If combining simulated configuration output is considered appropriate then all configurations are distributed to each client. A technical side note, all simulators discussed in this effort are designed to incorporate previously obtained data defined through runtime interpreted scripts.

Figure 7 highlights one example of partial job allocation. In this example, the server needs to allocate six jobs to three participating clients. Recall the server issues job commands, monitors job execution, and consolidates data from each client. Here, the configurations are designated A thru F. In partial job allocation, the server assigns jobs sequentially. In this example, client one is assigned jobs A and D for execution on workstation one. Note the assumption that jobs outnumber participating clients. Similar to second or later stages in ranking and selection methods, each configuration has different observational requirements. Observe the occasional idleness of workstations one and three.

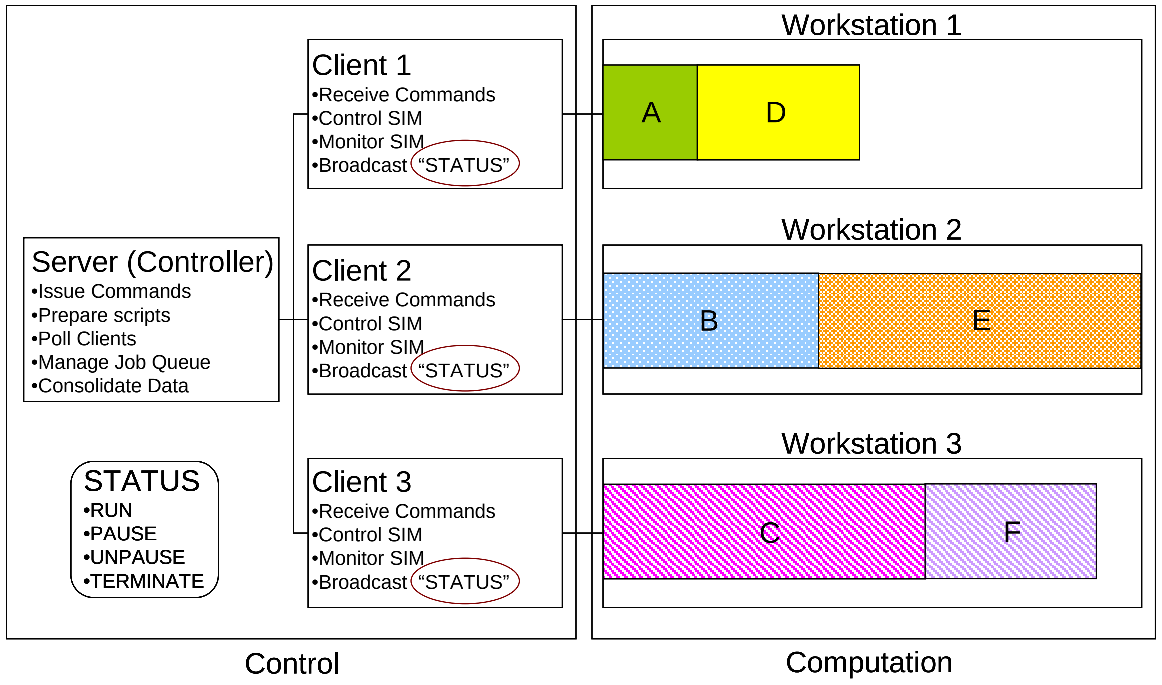


Figure 7: Example of Partial Job Allocation

Figure 8 highlights another method of job allocation. Again, the server needs to allocate six jobs of varying size to three participating clients. With full job allocation, each client is directed to execute an equal portion of all jobs. For example, if the observational requirement for job A is twelve then each client would contribute four observations. If the job cannot be equally divided then rounding up ensures adequate observation acquisition. Assuming the observational requirement is large negates the impact of these excess observations. However, the distribution of all jobs to all participating clients, assuming somewhat similar workstation performance, allows for near optimal execution by precluding idle time. Additionally, it avoids job transfer between workstations and the associated efficiency loss both in simulation initialization time and state variable data transfer.

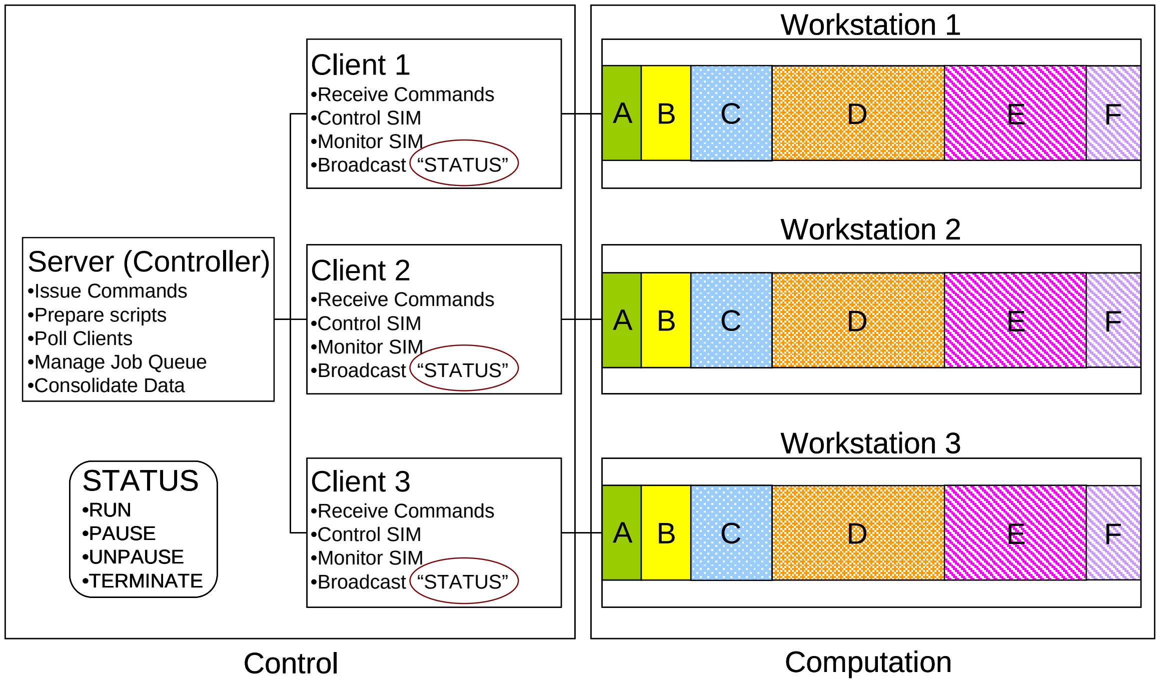


Figure 8: Example of Full Job Allocation

The central assumption enabling job allocation in this manner is the appropriateness of combining simulation output. Obviously, simulation output should only be combined from the same simulated configurations. The random number seeds must be different for the jobs, thereby ensuring observation independence. Also, combining output from one simulated configuration to another must not distort overall interpreted results or nullify inherent assumptions to the particular simulation. If combining simulation output cannot be done in this manner then this method of job allocation is problematic. Specific simulation requirements such as initialization time and the difficulty in transferring state space must be compared to the potential benefit of job rescheduling.

3.1.3 Distributed Simulation Exception Handling

The largest potential issue from this distributed simulation architecture is communication errors between the controller and participating workstations. Communication errors are handled by "loose" synchronization between the controller and participating workstations. Specifically, the controller can only discriminate between configurations when data required by ranking and selection methods are available. If data are not available for some or all configurations under contention, then the controller will pause for a specified period and subsequently reattempt data acquisition from participating workstations. Additionally, controller issued commands to participating workstations require confirmation of successful receipt. If this receipt is not obtained by the controller then the command is reissued after a specified period. Failed communication within the distributed simulation architecture is mitigated by these error handling techniques.

3.2 Distributed Simulation Performance

This section demonstrates performance of this distributed simulation architecture in a specific application. This sample experiment requires the selection of the "best" among six competing simulated configurations. For this experiment, the underlying process of the simulators is assumed to be an independent and identically distributed, *iid*, $Normal(\mu,\sigma) = N(\mu,1.5)$ distribution. The mean, μ, for five of the competing configurations was set to 0.0 while the mean for the sixth, or "best", configuration was set to 0.009682 for this experiment. Relatively large observational requirements are developed when batching methods, such as Batch Means (BM), are employed. Specifically, if the batch size, *m*, is increased when using BM, then the number of unbatched observations, n, also increases by the relationship $n = mb$ where b is the number of batches. For this performance demonstration the batch size $m = 1000$. The central issue here is the tradeoff between workstation performance and the overhead associated from the distributed simulation architecture. Specifically, contributing workstations should not be idle from a lack of controller issued commands, which generally results from slow communication, e.g., TCP/IP network bandwidth limitations.

Figure 9 highlights the distributed simulation architecture implemented on a homogeneous NOW comprised of dual Intel Xeon 2.2 GHz processors with 512 megabytes of RAM. Workstations communicated by operating system managed TCP/IP over a 100 megabit Ethernet connection for this experiment. Competing configurations of a normal *iid* process are compared using Rinott's procedure discussed in section 2.6. An experiment entails the selection of the "best" competing configuration. For this test case, 100 independent replications of the experiment facilitated estimation of the number of experiments completed per minute. In this test case, above-linear performance increases, in terms of the completed experiments, are obtained by the addition of more workstations. Note that each competing simulated configuration consumes computational resources if it is paused or actually generating observations. Distributing the computational requirement of a paused simulation along with observation requirements explains the above linear increase in performance.

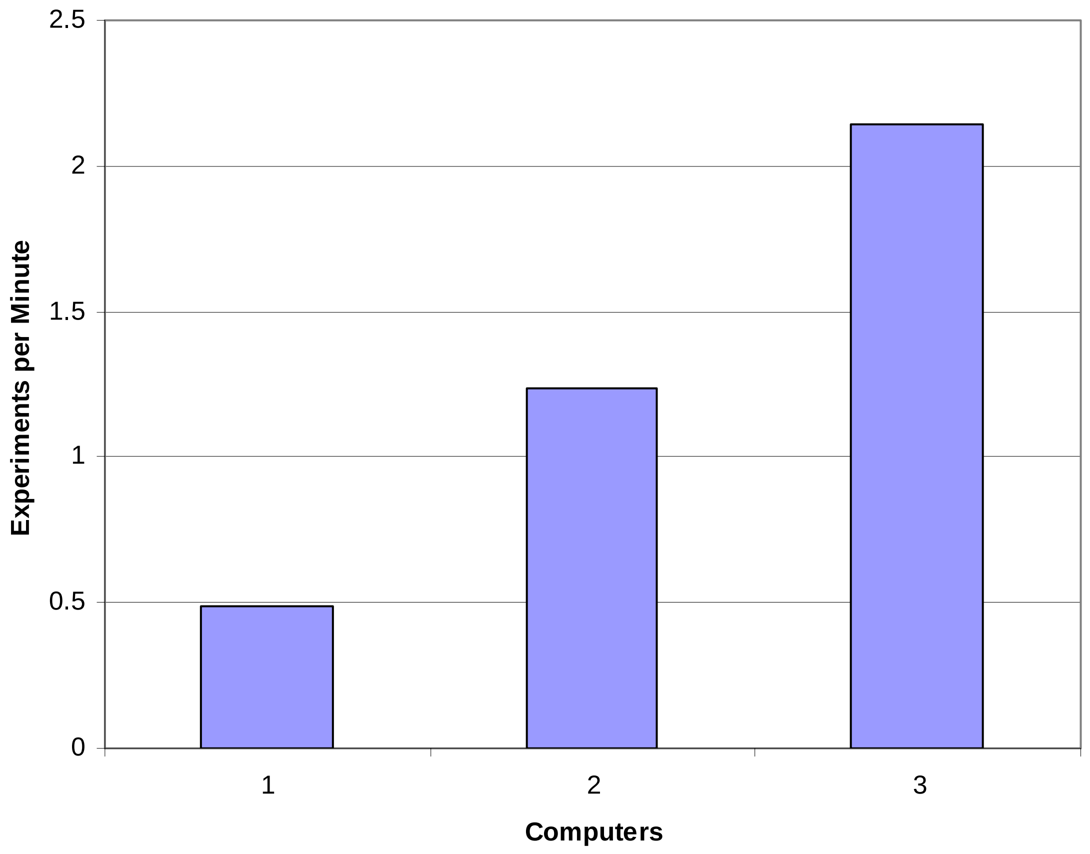


Figure 9: Test Case Performance

While small in comparison to observation acquisition, the controller for the distributed simulation architecture does consume computational capacity. So, the controller ran on an additional workstation to directly assess the computational impact of adding each additional workstation.

This experiment assessed the performance of a NOW comprised of up to four workstations. While not explored in this research, increasing the number of workstations contributing to an experiment will eventually result in a less than linear performance increase due to both network bandwidth and hard disk access limitations. However, this experiment is encouraging as it shows a small number of workstations may contribute computational capacity in a coordinated manner.

3.3 Summary

This distributed simulation architecture enables the efficient use of computational capacity for a small number of workstations. Comparison of differentiable simulated configurations facilitates distribution of computational requirements to participating workstations. Ranking and selection methods enable efficient calculation of the number of required observations and determination of which configurations are still in contention for selection as the “best”. Additionally, an assumption on the appropriateness of combining simulation output offers additional computational efficiency.

CHAPTER 4

RANKING AND SELECTION METHOD EXTENSION

This chapter focuses on the development, testing, and comparison of Ranking and Selection (RS) methods. RS methods enable efficient analysis of competing simulated configurations. Development of RS methods involves both theoretical analysis and empirical testing. Inherent RS assumptions and the specific goals of a method guide the theoretical analysis. Application of a RS method to simulations of an underlying normal or autoregressive process enables empirical testing. Together, this analysis and testing validates RS method performance.

Many of the test statistics required by ranking and selection methods have been made available at runtime by software developed for this thesis. Appendix B gives a brief example of test statistic calculations. Available test statistics, by dynamic link library (dll) access, include the multivariate normal, multivariate student t, studentized range distribution, and studentized maximum modulus distribution. Rinott's constant is also available. Previously, these test statistics were available from table lookups or from FORTRAN software (Bechhofer, Santner, and Goldsman, 1995).

4.1 Assumptions and Goals

The RS methods in this chapter obtain observations, X_{ij}, $j = 1,2,\ldots,$ for competing system configurations $i = 1,2,\ldots,k$ from either an independent, identically distributed normal, $iid - N(\mu,\sigma)$, process or from an autoregressive, $AR(1)$, process. RS methods described in section 2.6 assume observations are $iid - N(\mu,\sigma)$. The $AR(1)$ process facilitates analysis of more realistic simulation output that is serially correlated.

Note the use of batching methods on $AR(1)$ process data results in normal observations in certain conditions.

Requirements on whether the variance, σ_i^2, is known and/or equal can vary by the RS method. Generally, methods that allow for unknown/unequal variances require more observations to correctly select the “best” system configurations than methods assuming known/equal variance. All methods in this chapter will allow for unknown and unequal variances unless otherwise stipulated.

Using the indifference-zone formulation, the goal of RS methods is to select the system configuration with the “best”, e.g., largest, expected value, μ_i. The experimenter provides (δ^*, P^*), where δ^* is the indifference-zone parameter and P^* denotes the desired threshold probability of correctly identifying a difference between system configurations. Note that the indifference zone indicates some comparative region where the experimenter would not discriminate between system configurations. Also, the desired threshold probability, P^*, can be interpreted as a $1-\alpha$ confidence interval that configuration mean values do in fact differ by at least δ^*. Given *k* ordered means, $\mu_1, \mu_2, \ldots, \mu_k$, the probability requirement for this formulation is $P(CS) \geq P^*$ whenever $\mu_k - \mu_{k-1} \geq \delta^*$, where CS denotes correct selection.

4.2 Empirical Comparison Overview

There are several infrastructure requirements for comparing alternative RS methods. Each method must be parameterized in a similar manner to allow direct comparison. In this context, a parameterization denotes the selection of a simulated underlying process, indifference-zone parameter, desired probability, initial number of observations, and batch method along with its associated settings. In this controlled environment, a simulation mimicking either an $iid - N(\mu, \sigma)$ process or an $AR(1)$ process is required. The simulation architecture must also allow for both single and multi-stage RS methods, and must implement embedded data encapsulation in a manner that is both efficient and accurate. Lastly, metrics of method performance are necessary. This section details the techniques used in this effort to compare RS method performance.

4.2.1 Assessing Method Performance

The relative difference between competing system configurations directly impacts RS method performance. In the multivariate normal case where mean statistics $W_1, W_2, \ldots, W_k$ are obtained from k competing simulated system configurations with common correlation ρ, the equicoordinate multivariate normal point, $Z_{k-1,\rho}^{(1-P^*)}$, ensures compliance with the probability requirement:

$$P\left(\max_{1 \le i \le k} W_i \le Z_{k-1,\rho}^{(1-P^*)}\right) = P^* \tag{17}$$

The quantity $Z_{k-1,\rho}^{(1-P^*)}$ satisfies this probability requirement for any configuration of means in the form:

$$\mu_1 = \mu_{k-1} = \mu_k - \delta^* \tag{18}$$

This is often referred to as the *slippage* or Least Favorable (LF) configuration of means because of the strict equality induced in the probability requirement. Equal Spacing (ES) is another interesting configuration of means often used to compare RS method performance. For the ES configuration we will use $\mu_0 = 0, \mu_1 = \delta^*, \ldots, \mu_{k-1} = (k-1)\delta^*$ and $\mu_k = k\delta^*$ in our evaluations. The ES configuration of means relaxes the strict equality in the probability requirement and for such competing system configurations, it is usually easier to distinguish the “best”. All RS methods in this chapter are applied to a LF configuration of means unless otherwise noted.

The ratio of δ^*/σ also impacts RS method performance. Recall the indifference-zone parameter, δ^*, is a comparative region where the experimenter would not discriminate between competing system configurations. Also, RS methods discussed in this chapter assume unknown variance. However, a controlled environment, enabled by the simulation of an $iid - N(\mu,\sigma)$ process or an $AR(1)$ process, facilitates performance evaluation of RS methods. Here, the asymptotic variance is known or can be estimated, thereby allowing manipulation of δ^* for RS method performance evaluation purposes. If the ratio is “too small”, then the number of required observations can be prohibitively high. If the ratio is large, then it is it difficult to differentiate between the performance of RS methods as all will have modest sample-size requirements.

Knowledge of the asymptotic variance of the underlying simulated process allows for good selection of the ratio δ^*/σ. For example, if it is assumed that 24 initial observations, n_0, is an adequate sample size for obtaining relatively good variance estimation, then generating random numbers from a $Normal(\mu,\sigma) = N(0,1.5)$ distribution with batch size $m = 1$ allows selection of the indifference-zone parameter in the following manner:

$$\delta^* = \sigma/\sqrt{n_0} = 1.5/\sqrt{24} = 0.30619 \qquad (19)$$

Selection of this parameter enables analysis of performance within one standard deviation of prescribed performance. All RS method comparisons in this chapter select the indifference-zone parameter, δ^*, in a similar fashion.

Given similar parameterization, 1000 independent experiments have been replicated of the RS methods given here to empirically assess performance. Unless otherwise noted, the LF configuration of means is used where $N(\mu_i, 1.5)$ random numbers are generated with $\mu_1 = \mu_{k-1} = 0$, $\mu_k = 0.30619$, $k = 6$, $m = 1$, and $\delta^* = 0.30619$. In this context, an experiment denotes the use of a particular RS method to determine the "best" of $k = 6$ system configurations. Here, the "best" system configuration is $\mu_6 = 0.30619$. Note that an $iid - N(\mu, \sigma)$ process with batch size, $m = 1$, is used for initial RS method comparison and development. This simplification eases computational expense and analysis. However, batching techniques such as Batch Means (BM) and Overlapping Batch Means (OBM) are applied later in this chapter to an $AR(1)$ process to assess RS method robustness to serially correlated simulation output.

Separate simulators, in terms of data storage and parameterization, are used for each system configuration. Each simulator provides either $iid - N(\mu, \sigma)$ or $AR(1)$ observations following parameters set at runtime through the use of script files. At each stage each simulator has the ability to communicate its status and interpret controller issued commands, thus enabling RS methods to be applied by the controller module. Using the controller, each experiment automatically terminates and regenerates until the required number of experiments are replicated. Also, the controller module stores experimental outcomes that summarize the performance of the RS method after each experiment replication. A detailed discussion of the control architecture can be found in section 3.1.

After a RS method has been implemented, there are two performance metrics that facilitate side-by-side comparison. First, the achieved probability of correct selection, $P(CS)$, indicates whether the method meets or exceeds the desired probability P^*. For the indifference-zone formulation, the event of correct selection is observed when the "best" configuration is in fact selected by the method. The second metric for comparing RS methods involves the average number of required raw or unbatched observations, $\overline{T}$, necessary to select a configuration. This metric corresponds to the computational efficiency of the method. All RS method comparisons in this chapter utilize these metrics to assess performance and computational requirements.

4.2.2 Data Encapsulation Methods

Embedded estimators of mean and variance enable RS method calculations, such as the number of required observations. Point estimators for the mean are relatively easy to calculate as they sum observation values, in this case X_i, and divide by the number of observations, here *n*:

$$\overline{X} = \sum_{i=1}^{n} \frac{X_i}{n} \tag{20}$$

Variance estimation of this sample mean is found from sample observations by:

$$V\hat{A}R\left[\overline{X}\right] = \frac{S^2}{n} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n(n-1)} \tag{21}$$

By algebraic manipulation:

$$\hat{VAR}\left[\overline{X}\right] = \frac{n\sum_{i=1}^{n} X_i^{\,2} - \left(\sum_{i=1}^{n} X_i\right)^2}{n^2(n-1)} \tag{22}$$

Selected terms from the last relationship can be calculated and stored during simulator execution thereby precluding the need for storage of historical data.

Under certain conditions it may be necessary to combine simulation output, as discussed in section 2.2.3. The following relationship enables the combination of variance estimators:

$$\sum_{i=1}^{n} X_i^{\,2} = \frac{n^2(n-1)\hat{VAR}\left[\overline{X}\right] + \left(\sum_{i=1}^{n} X_i\right)^2}{n} \tag{23}$$

Also, if using batch means of size *m* with independent and identically distributed normal data, then variance estimators are related in the following manner:

$$\hat{VAR}\left[\overline{X}\right]_{Batch} = \frac{\hat{VAR}\left[\overline{X}\right]_{Obs}}{m} \tag{24}$$

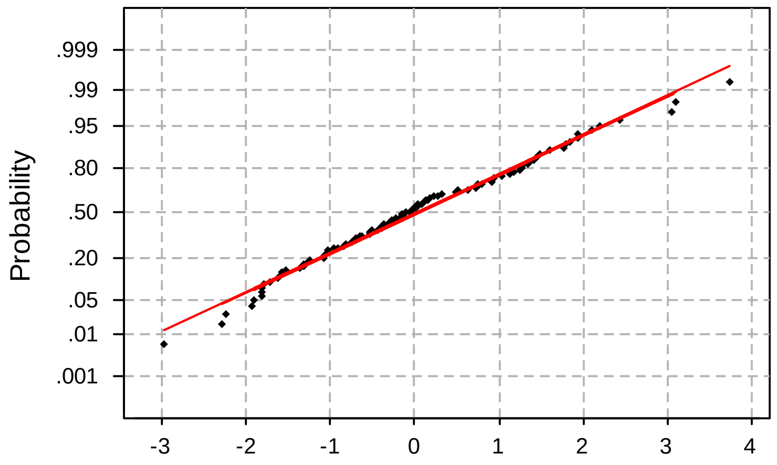
Technical requirements for using these embedded estimators reside on ensuring the absence of under/overflow within developed software. In particular, the $\sum_{i=1}^{n} X_i^{\,2}$ term can become relatively large. Also, there are known issues with C, C++, and C# when variable typecasting is absent.

4.2.3 Random Number Generator Verification

Simulators for both $iid - N(\mu,\sigma)$ and $AR(1)$ processes require random number generators. The uniform random number generator used here is the multiple recursive generator presented in Law and Kelton (2000). The normal random number generator is the polar Acceptance-Rejection (A-R) method described in the same source. The normal random number generator provides the underlying process for the simulator that uses stipulated configuration parameters set at runtime to include the initial random number, or seed.

Implementation of a known random number generator on any compiler requires some form of empirical testing. Specific implementation issues generally revolve on correctly mimicking the prescribed distribution, serial correlation of the data, and the relative independence of observed data. To that end, empirical testing of the implemented normal random number generator follows.

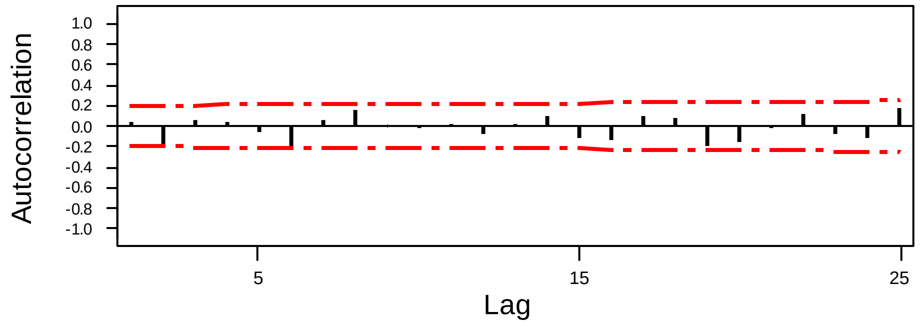
The probability plot shown in Figure 10 indicates a relatively good $Normal(\mu,\sigma) = N(0,1.5)$ distribution. The Anderson-Darling test value is high, thereby reinforcing confidence that the generator is in fact performing properly. The small serial correlation, or in this case autocorrelation if observations are assumed to be time-based, shown in Figure 11 is also promising. A runs test on the data also indicates independence. Incorporated into a basic simulator, these results verify random number generator characteristics enabling performance comparison between different RS methods.



```
Av erage: 0.0261289                                  Anderson-Darling Normality  Test
StDev : 1.31281                                               A-Squared: 0.399
N: 100                                                        P-Value:    0.359
```

Figure 10: Polar Acceptance-Rejection Normal Random Number Generator Probability Plot

and,



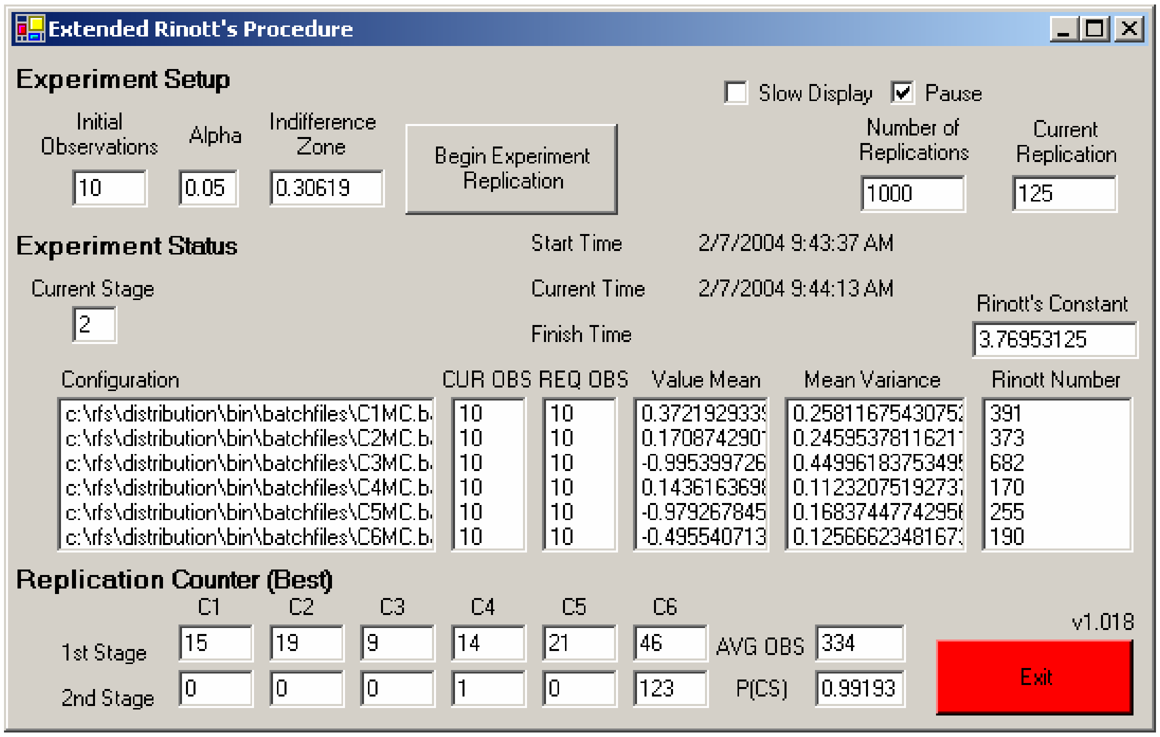
```
Lag  Corr     T    LBQ    Lag  Corr     T    LBQ    Lag  Corr     T    LBQ    Lag  Corr     T    LBQ

  1  0.03  0.35   0.13      8  0.16  1.52  11.92     15 -0.13 -1.14  16.14     22  0.12  0.95  29.51
  2 -0.18 -1.81   3.55      9  0.00  0.00  11.92     16 -0.13 -1.13  18.15     23 -0.07 -0.57  30.17
  3  0.06  0.59   3.94     10 -0.02 -0.22  11.99     17  0.10  0.87  19.37     24 -0.12 -0.95  32.01
  4  0.04  0.34   4.07     11  0.03  0.25  12.07     18  0.07  0.62  20.01     25  0.18  1.45  36.44
  5 -0.07 -0.65   4.57     12 -0.09 -0.77  12.92     19 -0.20 -1.70  24.97
  6 -0.19 -1.82   8.48     13  0.03  0.25  13.02     20 -0.15 -1.23  27.75
  7  0.06  0.60   8.94     14  0.10  0.89  14.18     21 -0.01 -0.11  27.77
```

Figure 11: Polar Acceptance-Rejection Normal Random Number Generator Autocorrelation

4.2.4 Sample RS Method Experiment

Given a simulator mimicking a system with an underlying $iid - N(\mu,\sigma)$ or $AR(1)$ process, a control mechanism is required to implement RS methods. Communication of simulated system data at each stage, i.e., obtaining a specified number of observations, enables the application of RS methods. Here, an experiment consists of using a RS method to select the "best" system configuration. The following sample experiment highlights the specific simulation architecture and process used in this effort.

Figure 12 highlights an implementation using this control mechanism to use Rinott's method (discussed in section 2.6). Recall Rinott's method is two-stage. Experimental setup, in the top left, includes RS method parameterization of the first-stage number of observations $n_0 = 10$, the desired probability $P^* = 1 - \alpha = 0.95$, and the indifference-zone parameter $\delta^* = 0.30619$. The underlying process for this sample experiment is $iid - N(\mu,\sigma)$. Shown in the top right, the experimenter has set the required number of replications to 1000. At the time of the snapshot in Figure 12, the control mechanism is between the first and second-stage of experiment 125 out of the required 1000 experiments. Under experimental status, observe approximately 36 seconds of computer-time have elapsed. Rinott's constant is an integral component of this RS method. The location of simulation initialization files helps to identify competing configurations. Estimated mean and variance highlight specific system configuration performance. Of interest, the "Rinott Number" is the total number of observations estimated by the RS method to be necessary for system configuration comparison. Note that higher first-stage variability results in a higher number of estimated raw observations. Overall performance of the method is described by estimators of the probability of correct selection (CS), $\hat{P}(CS)$, and the average number of required unbatched observations, $\hat{\bar{T}}$. In this case, the performance estimators are 0.992 and 334 respectively. Knowledge of the true "best" configuration allows for calculation of $\hat{P}(CS)$.


Extended Rinott's Procedure

Experiment Setup

Initial Observations: 10
Alpha: 0.05
Indifference Zone: 0.30619
Begin Experiment Replication
Slow Display ☐ Pause ☑
Number of Replications: 1000
Current Replication: 125

Experiment Status

Current Stage: 2
Start Time: 2/7/2004 9:43:37 AM
Current Time: 2/7/2004 9:44:13 AM
Finish Time:
Rinott's Constant: 3.76953125

Configuration	CUR OBS	REQ OBS	Value Mean	Mean Variance	Rinott Number
c:\rfs\distribution\bin\batchfiles\C1MC.b	10	10	0.372192933	0.258116754307	391
c:\rfs\distribution\bin\batchfiles\C2MC.b	10	10	0.170874290	0.245953781162	373
c:\rfs\distribution\bin\batchfiles\C3MC.b	10	10	-0.995399726	0.449961837534	682
c:\rfs\distribution\bin\batchfiles\C4MC.b	10	10	0.143616369	0.112320751927	170
c:\rfs\distribution\bin\batchfiles\C5MC.b	10	10	-0.979267845	0.168374477429	255
c:\rfs\distribution\bin\batchfiles\C6MC.b	10	10	-0.495540713	0.125666234816	190

Replication Counter (Best)

	C1	C2	C3	C4	C5	C6
1st Stage	15	19	9	14	21	46
2nd Stage	0	0	0	1	0	123

AVG OBS: 334
P(CS): 0.99193
v1.018
Exit

Figure 12: Sample Ranking and Selection Method Experiment

Comparing first and second-stage counters of experiment replications in which each system configuration is considered the "best" at a particular stage requires explanation based on knowledge of the system configurations. In this example, all of the system configurations were considered the "best" during the first-stage of at least one of the 124 initial experiments, but not necessarily the final choice as "best". This implies the system configurations are closely competitive, unless enough observations are taken.

Observe the last system configuration, C6, is selected as the “best” in 99% of the experiments.

4.3 R+ and KN+ Methods Performance Analysis

Comparison of the Rinott (R+) and the Kim and Nelson (KN+) methods allows for verification and validation of the implementation, and also provides insight for improving RS methods. Verification is obtained by manual numerical comparison using spreadsheets and table lookups. Validation comes from comparing performance trends of these methods to other published analyses. Insight for new method development comes from both analysis of these methods’ algorithms and their observed performance.

Initially, both methods are parameterized in the ES and LF configuration of means with $\delta^* = 0.30619$, $k = 6$, $m = 1$, and an $iid - N(\mu, \sigma)$ underlying process. After method performance comparison on ES and LF configurations, the LF configuration will be used primarily unless otherwise noted. Comparative analysis focuses on varying the desired probability P^*, the first-stage number of observations n_0, and ultimately the batch size m. Performance metrics are estimators of the probability of correct selection, $\hat{P}(CS)$, and average number of required raw/unbatched observations, $\hat{\bar{T}}$, obtained from 1000 independent experiment replications. Since the KN+ method is multistage, the upper bound on the number of required unbatched observations, determined at the end of the first stage as detailed earlier in section 2.6.2, is also reported. The remainder of this section discusses experimental results.

4.3.1 Varying Desired Probability

In this experiment, ranking and selection method performance on both the LF and ES configuration of means is explored as the desired probability P^* is varied. As shown in Table 1 and Figure 13, both R+ and KN+ methods exceed the desired probability in all conditions. In fact, the methods surpass the desired probability implying the number of required observations is higher than necessary. KN+ outperforms R+ in exceeding the desired probability at all levels. Also, note the desired probability is exceeded to a greater extent in the ES configuration of means than the LF configuration. This follows since it is easier to distinguish between competing populations in the ES configuration than in the LF configuration. For the KN+ method, the estimated observation requirements along with the associated upper bound calculated by this method are reported.

Table 1: R+ and KN+ Comparison Varying Desired Probability

$\delta^* = 0.30619$, $n_0 = 10$, **LF/ES**, $iid - N(\mu,\sigma)$, $m = 1$					
		LF		**ES**	
P^*		$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
0.75	R+	0.815	146	0.934	145
	KN+	0.886	103/349	0.955	57/348
0.90	R+	0.977	265	0.978	251
	KN+	0.984	169/568	0.992	96/568
0.95	R+	0.968	337	0.990	337
	KN+	0.976	236/754	0.993	131/746

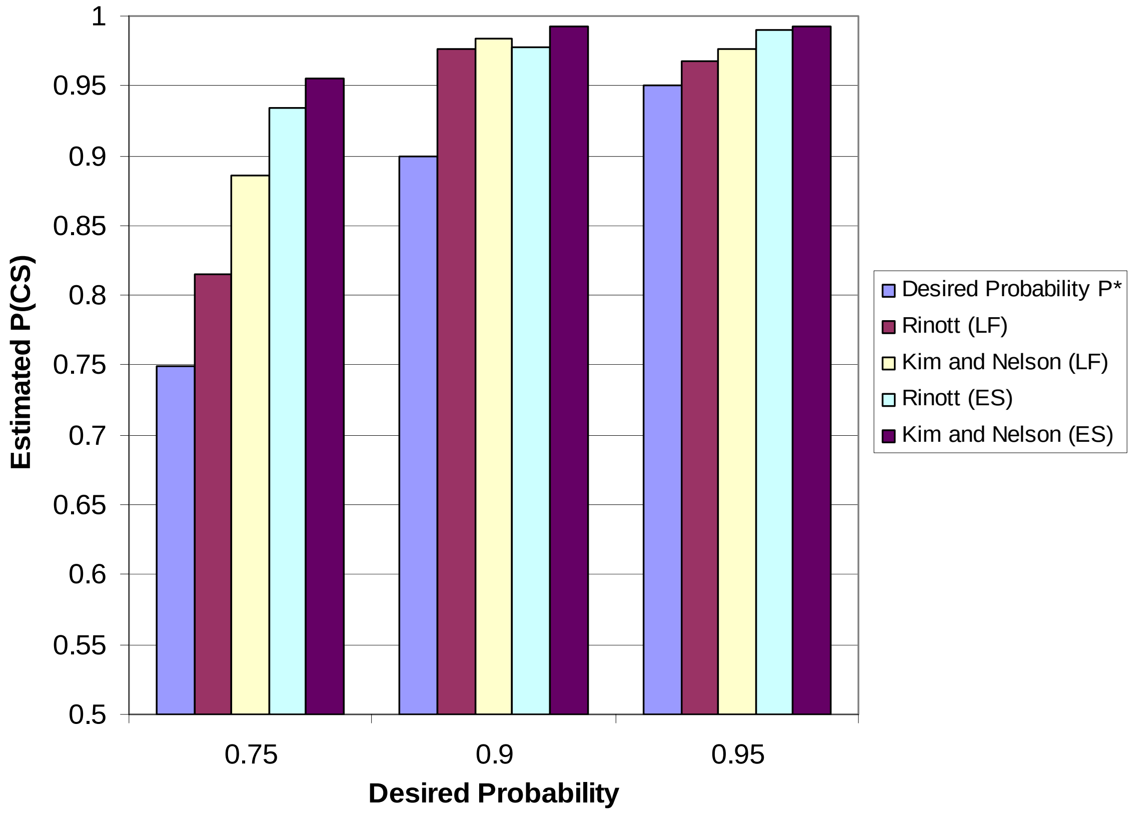


Figure 13: R+ and KN+ Estimated P(CS) versus Desired Probability

The average number of required raw observations, $\hat{\bar{T}}$, shown in Figure 14, increases as the desired probability, P^*, is raised for both methods. From the tabular values, observe the upper bound on the estimated number of required observations for KN+ is much higher than the number of required observations for R+. On the other hand, the benefit of the multi-stage nature of the KN+ method is shown by directly comparing the number of required observations. Unlike the two-stage R+ method, the multi-stage aspect of the KN+ method allows for elimination of simulated system configurations, resulting in a lower average number of required raw observations. Also, the average number of required observations is smaller for the ES mean configuration than the LF mean configuration. Again, it is easier to distinguish between competing ES configurations than those in the LF configuration.

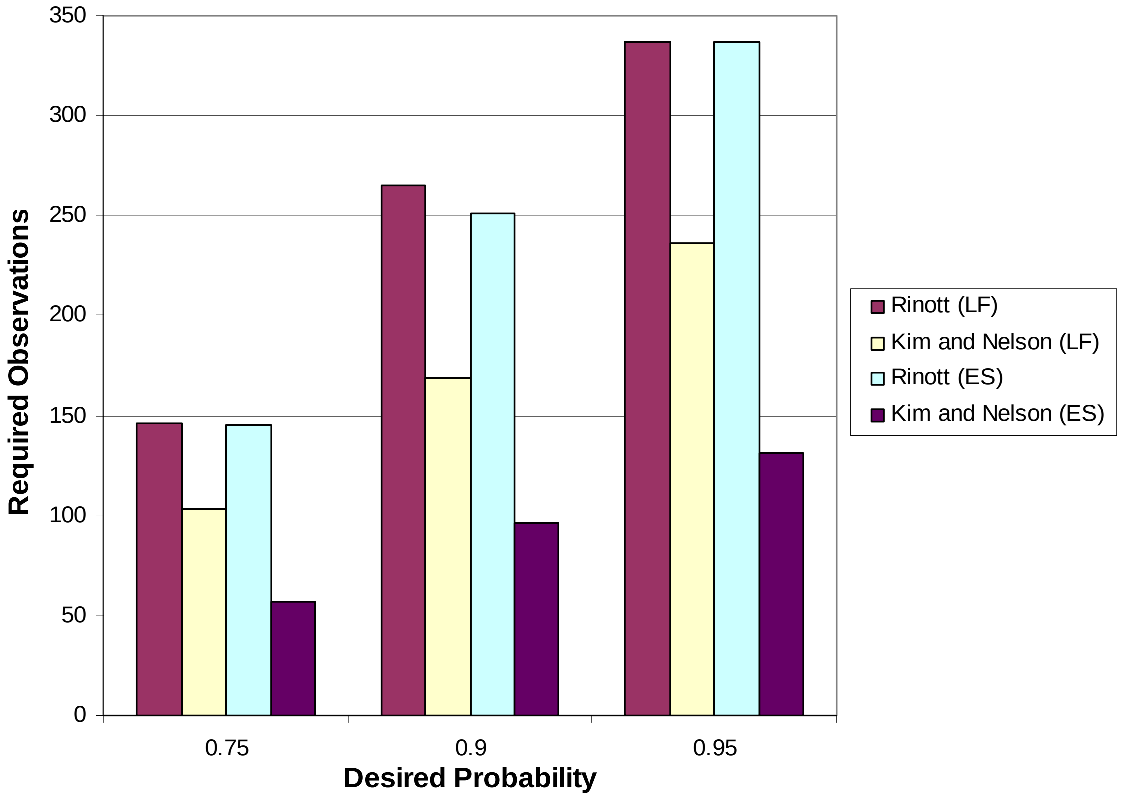


Figure 14: R+ and KN+ Required Observations versus Desired Probability

4.3.2 Varying First-Stage Number of Observations

In this experiment, the initial number of observations, n_0, was varied. Parameterization for this experiment includes an underlying $iid - N(\mu, \sigma)$ process, batch size $m = 1$, indifference-zone parameter $\delta^* = 0.30619$, $k = 6$ competing system configurations, and a desired probability $P^* = 0.95$ in the LF configuration of means. As shown in Table 2 and Figure 15 below, the KN+ method requires fewer total raw observations, $\hat{\bar{T}}$, than the R+ method except when n_0 is large (where both methods require the same amount). Achieved $\hat{P}(CS)$ is statistically equivalent to or exceeds the desired probability, P^*, in all conditions. Also, a large number of initial observations creates computational inefficiency, i.e., a large total observation requirement, in both methods. While the upper bound for required observations for the KN+ method is always larger than that for the R+ method, the screening process within the KN+ method allows for increased computational efficiency. This efficiency is obtained by eliminating competing system configurations during the screening phase of this method, as discussed in section 2.6.2.

Table 2: R+ and KN+ Comparison Varying Initial Number of Observations

$\delta^* = 0.30619$, $P^* = 0.95$, **LF**, $iid - N(\mu,\sigma)$, $m=1$							
n_0	Method	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	n_0	Method	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
8	R+	0.982	420	100	R+	0.951	262
	KN+	0.990	277/928		KN+	0.968	143/419
10	R+	0.990	356	110	R+	0.963	263
	KN+	0.968	235/764		KN+	0.984	147/416
20	R+	0.952	292	120	R+	0.969	263
	KN+	0.968	162/533		KN+	0.978	152/416
30	R+	0.951	278	130	R+	0.967	263
	KN+	0.971	152/481		KN+	0.943	157/413
40	R+	0.967	274	140	R+	0.986	261
	KN+	0.992	149/468		KN+	0.986	162/409
50	R+	0.970	269	150	R+	0.955	261
	KN+	0.963	141/446		KN+	0.969	168/405
60	R+	0.976	258	200	R+	0.951	260
	KN+	0.957	137/420		KN+	0.993	205/399
70	R+	0.971	266	250	R+	0.963	262
	KN+	0.975	140/432		KN+	0.969	252/395
80	R+	0.948	262	300	R+	0.966	300
	KN+	0.967	141/428		KN+	0.979	301/394
90	R+	0.958	261	400	R+	1.000	400
	KN+	0.972	143/422		KN+	1.000	400

Figure 15 shows how the initial number of observations impacts the total number of observations for both methods. Both methods exhibit concave behavior where both a low and high number of initial observations, n_0, equate to a high number of total raw observations. The location of this curve relative to the number of initial observations, depends upon the selection of the indifference-zone parameter. So, if there is no fore-knowledge on the variance of the underlying process, as assumed in both the R+ and KN+ methods, then selecting the number of initial observations can be problematic.

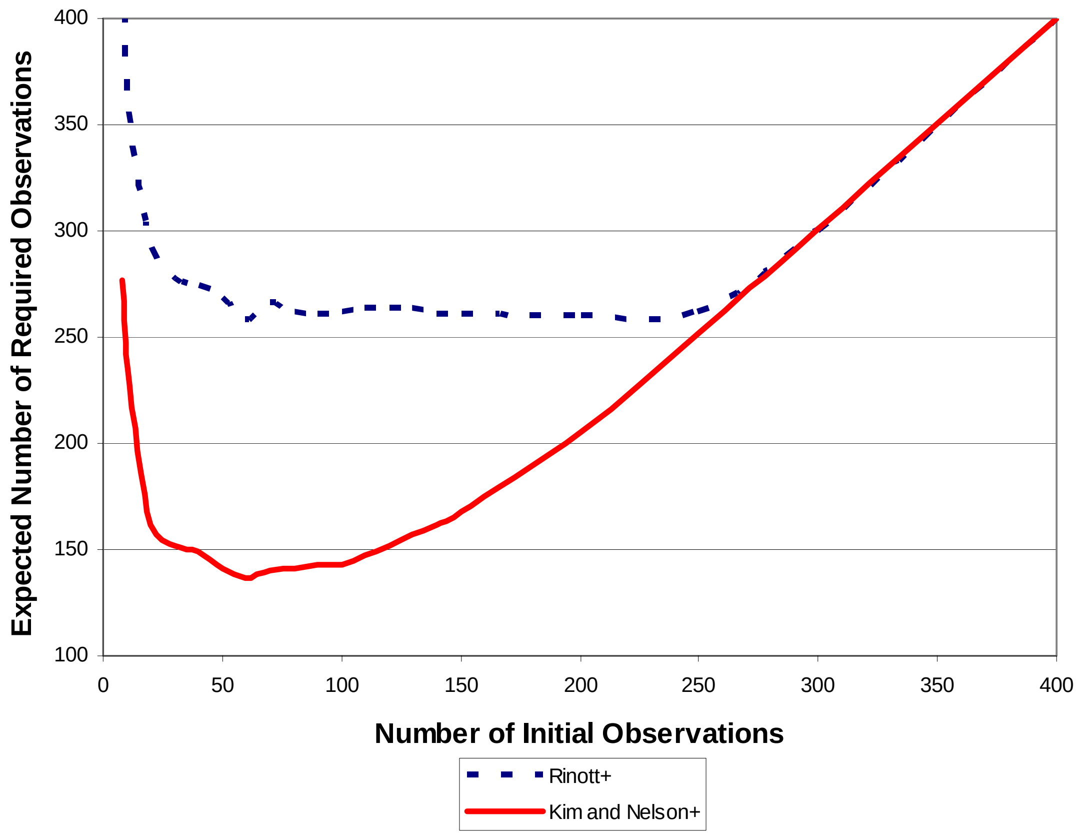


Figure 15: Initial Number of Observations versus Required Number of Observations

4.3.3 Batched Data Method Performance

The RS methods discussed in section 2.6 rely on the assumption of independent and identically distributed, *iid*, normal data. Batching methods, discussed in section 2.5, enable approximately *iid* normal observations from underlying non-normal distributions when *m*, the batch size, is sufficiently large. The batching methods explored in this section include Batch Means (BM) and Overlapping Batch Means (OBM). Incorporation of these batching methods permits the application of RS methods to more realistic simulations that generate data from a variety of stochastic processes. As a test case, batching methods are applied to an autoregressive $AR(1)$ process that mimics a system with correlated observations. Note an $AR(1)$ process is often used to represent observations from a time-based system. With a mean for system *i*, u_i, an $AR(1)$ process generates each observation X_{ij}, $j = 1,2,\ldots,$ for competing system configurations $i = 1,2,\ldots,k$ from the relationship:

$$X_{i,j} = u_i + \phi(X_{i,j-1} - u_i) + Z_{i,j} \tag{25}$$

$R_k = Cov(X_{i,j}, X_{i,j-k}) = \phi^k$, where $-1 < \phi < 1$. The error terms, $Z_{i,j}$, are distributed *iid* $N(0,1-\phi^2)$.

Unless specified otherwise, $\phi = 0.22$, which creates mildly correlated observations. A variance estimator for correlated data follows:

$$m\hat{VAR}\left[\overline{X}\right]_{Batch} = R_0 + \sum_{k=1}^{m-1}\left(1 - \frac{k}{m}\right)R_k \tag{26}$$

It can be shown that with large batch size, *m*, the variance of the sample mean for an $AR(1)$ process converges to:

$$mVAR\left[\overline{X}\right]_{Batch} \rightarrow \frac{1+\phi}{1-\phi} \tag{27}$$

Using $\hat{\sigma}^2$ as an estimator for $m\hat{VAR}\left[\overline{X}\right]_{Batch}$ facilitates our choice of the indifference-zone parameter as:

$$\delta^* = \hat{\sigma}\Big/\sqrt{mn_0} = 1.251\Big/\sqrt{n_0} \tag{28}$$

The remainder of this section focuses on R+ and KN+ technique performance using an $AR(1)$ process with BM and OBM data acquisition methods.

Batch Means

The BM method obtains *b* batched observations of size *m*. The number of initial batches may be obtained from the relationship $b_0 = \lceil n_0/m \rceil$ where n_0 is the number of initial unbatched or raw observations. A side note, the embedded data estimators create batched observations for the adaptive control techniques given in section 2.6, requiring no more than $m-1$ excess unbatched observations from the simulation. Central to any batching method is how large the batch size must become to enable sufficient estimation of the underlying process variance, $\sigma^2 \equiv \lim_{m\rightarrow\infty} mVAR\left[\overline{X}\right]$. For any process the convergence of $mVAR\left[\overline{X}\right]$ to σ^2 may be demonstrated by simulating the process while increasing the batch size as long as the underlying distribution is stationary (along with other mild conditions).

For example, the following experiment illustrates variance convergence for a specific $AR(1)$ process. Figure 16 highlights estimated variance using the BM method with an $AR(1)$ process with $\phi = 0.22$ and $n_0 = 20000$ as a function of batch size. 1000 independent replications of a simulated $AR(1)$ process facilitated asymptotic variance estimation. Note asymptotic variance estimation is obtained by averaging the variance estimator from each experiment. In fact, variance estimators have inherent variability as a result of the underlying $AR(1)$ process. As both R+ and KN+ methods rely on variance estimators to determine the number of required observations, underestimation of variance will result in a lower estimate of required observations, with a corresponding lower $P(CS)$. For this particular $AR(1)$ process, batch sizes below 40 result in underestimation, at some points significant, of the asymptotic variance. Batch sizes above 40 indicate sufficient convergence of the variance estimators to the asymptotic variance. Note the asymptotic variance is indicated by the flat line obtained from the relationship $\frac{1+\phi}{1-\phi} = 1.564$.

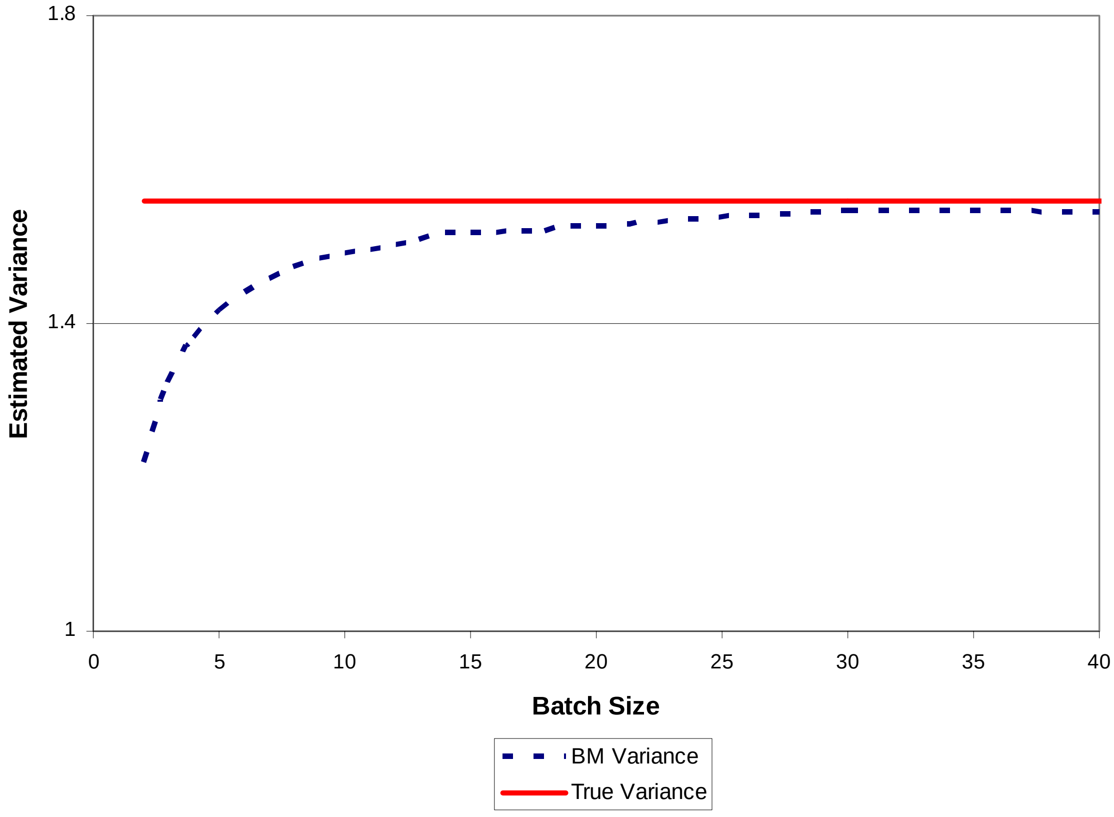


Figure 16: BM Estimated Variance Parameter versus Batch Size for AR(1) Data

Table 3 presents the experimental results of the R+ and KN+ methods applied to an $AR(1)$ process while obtaining observations with the BM method. Experiment parameterization involved setting $\phi = 0.22$, $n_0 = 4200$, $P^* = 0.95$, $k = 6$ competing system configurations, and $\delta^* = 0.019298 = \sqrt{1.564/4200}$ while varying both the batch size, *m*, and the initial number of batched observations b_0. The required raw observation upper bound for the KN+ method is also reported. Intuitively, as b_0 decreases, the number of required unbatched or raw observations, $\hat{\bar{T}}$, increases. Since the variance estimator is based on a χ^2 distribution with $b_0 - 1$ degrees of freedom, the variance of that distribution is high for a low b_0. This is consistent with results found in the *iid* case.

Of special interest is the relatively poor performance, in terms of achieving the desired probability, of the R+ method when the batch size is small. This can be attributed to the lack of asymptotic convergence of the variance estimator. An experiment follows to determine a sufficiently large batch size for acceptable R+ method performance. Note that, in this experiment, the KN+ method is not as susceptible to poor $\hat{P}(CS)$ performance as the R+ method when there is a lack of asymptotic variance convergence. Lastly, KN+ requires far fewer raw observations due to its screening process.

Table 3: R+ and KN+ Comparison Using Batch Means while Varying Batch Size

$\delta^* = 0.019298$, $P^* = 0.95$, $n_0 = 4200$, **LF**, $AR(1)$, $\phi = 0.22$					
		R+		**KN+**	
m	b_0	$\hat{P}(CS)$	$\hat{\bar{T}}$	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
10	420	0.887	43244	0.968	20912/64676
25	168	0.940	44861	0.964	22834/69939
50	84	0.936	45917	0.972	23005/73263
100	42	0.952	46825	0.976	25219/82739
150	28	0.976	51321	0.955	30191/94992
200	21	0.956	51826	0.974	29919/99284
300	14	0.941	53732	0.962	33371/110908

Table 4 explores the R+ method's $\hat{P}(CS)$ performance as the number of initial unbatched observations is increased. This experiment determines if this specific $AR(1)$ process with $\phi = 0.22$, a batch size of 300 or larger and $n_0 > 4200$ ensures sufficient variance convergence for acceptable R+ method performance. Parameterization for this experiment includes $P^* = 0.95$, $m = 300$, $k = 6$ competing system configurations, and $\delta^* = \sqrt{1.564/n_0}$ varying with the number of initial unbatched observations. Compared to the previous experiment, this experiment indicates the R+ method achieves the desired probability with a sufficiently large batch size. However, the number of unbatched observations is significantly higher thereby increasing computational expense.

Table 4: R+ Method Analysis with Varying Initial Unbatched Observations

$P^* = 0.95$, $m = 300$, $\delta^* = \sqrt{1.564/n_0}$, **LF**, $AR(1)$, $\phi = 0.22$			
n_0	b_0	$\hat{P}(CS)$	$\hat{\hat{T}}$
4200	14	0.941	53732
8400	28	0.973	95148
12600	42	0.964	140036
16800	56	0.957	189987
21000	70	0.963	229749
25200	84	0.984	280430

Overlapping Batch Means

The Overlapping Batch Means (OBM) method obtains $b = n - m + 1$ batched observations. The number of initial batches may be obtained from the relationship $b_0 = n_0 - m + 1$ where n_0 is the number of initial unbatched or raw observations. Again, the embedded data estimators create batched observations for the adaptive control techniques given in section 2.6. A side note, to speed the distributed simulation architecture discussed in section 3.1, simulation sampling was modified to acquire m OBM observations for multi-stage RS methods. Worst case from this modification is $m-1$ excess unbatched observations from the simulation executable.

Clearly, for the same number of unbatched or raw observations OBM obtains more batched observations than BM; however, OBM batches are highly correlated. It can be shown that as both the ratio $b = n/m$ and m become sufficiently large, the following estimator is consistent for the underlying process variance:

$$mV_0^2 \equiv \frac{nm}{(n-m+1)(n-m)} \sum_{j=1}^{n-m+1} \left(\overline{X}_i(j,m) - \overline{X}_i(n) \right)^2 \qquad (29)$$

Figure 17 highlights the estimated variance of an $AR(1)$ process with $\phi = 0.22$ and $n_0 = 20000$ while varying batch size using the OBM method. This experiment determines the location of asymptotic variance convergence for this specific $AR(1)$ process. 1000 independent replications were made at selected n/m ratios. Note asymptotic variance estimation is obtained by averaging the variance estimator from each experiment. Again, as in the BM case, variance estimators have inherent variability as a result of the underlying $AR(1)$ process. As both the R+ and KN+ methods rely on variance estimators to determine the number of required observations, underestimation of variance will result in a lower estimate of the number of required observations, with a corresponding lower $P(CS)$. This empirical analysis implies a ratio n/m greater than 8 is necessary for OBM usage when applied to an underlying $AR(1)$ process with $\phi = 0.22$.

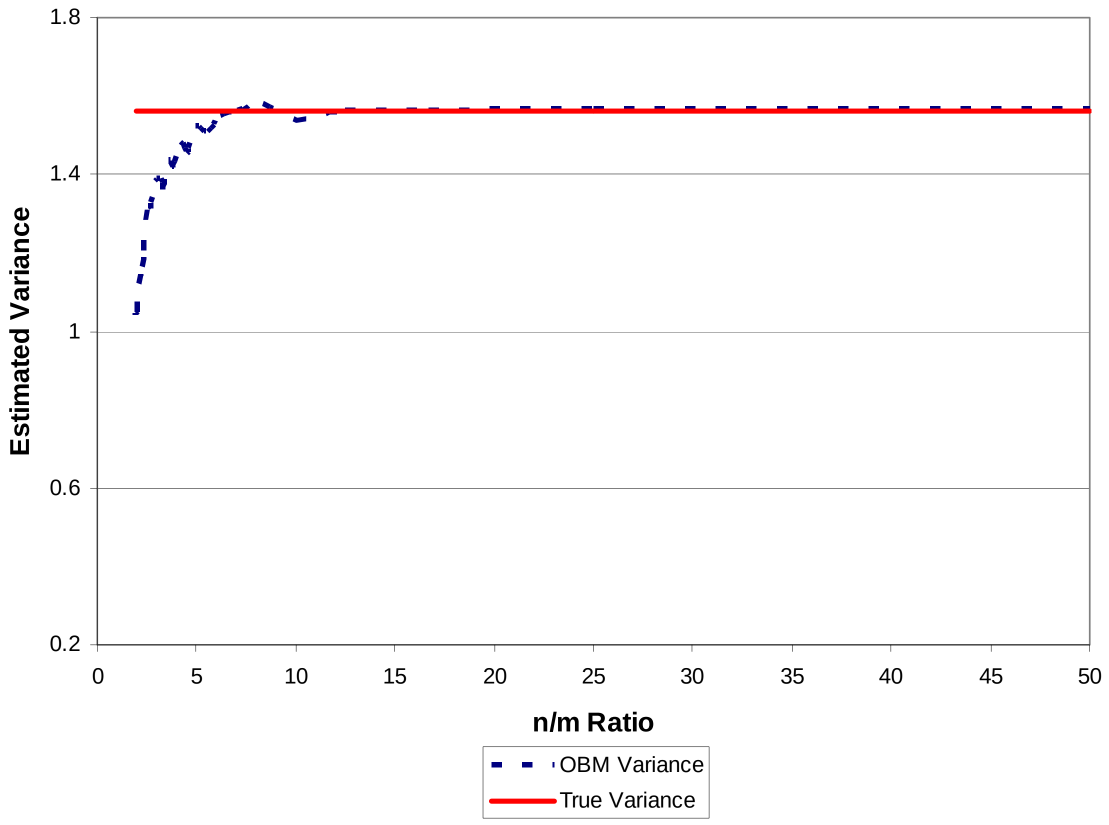


Figure 17: OBM Estimated Variance versus n/m Ratio for AR(1) Data

Table 5 presents the experimental results of the R+ and KN+ methods applied to an $AR(1)$ process while obtaining batched observations with the OBM method. Parameterization for this experiment includes $\phi = 0.22$, $n_0 = 8400$, $P^* = 0.95$, $k = 6$ competing system configurations, and $\delta^* = 0.013646 = \sqrt{1.564/8400}$; both batch size, *m*, and the initial number of OBM observations, b_0, were varied. Note the number of initial unbatched observations, n_0, remains constant.

This experiment highlights the necessity of asymptotic variance convergence for appropriate use of the R+ and KN+ ranking and selection methods. Asymptotic variance estimator convergence for an $AR(1)$ process is obtained by both sufficiently large *m* and a large n/m ratio when using OBM. Ranking and selection method performance is relatively poor in this experiment indicating *m* and/or the ratio n/m are not sufficiently large. Observe that the estimated $\hat{P}(CS)$ is nominally achieved with a large n/m ratio, implying a necessity for an increase in the number of initial unbatched observations.

Table 5: R+ and KN+ Comparison Using Overlapping Batch Means while Varying Batch Size, $n_0 = 8400$

$\delta^* = 0.013646$, $P^* = 0.95$, $n_0 = 8400$, **LF**, $AR(1)$, $\phi = 0.22$					
		R+		**KN+**	
m	b_0	$\hat{P}(CS)$	$\hat{\bar{T}}$	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
10	8391	0.951	86819	0.953	41872/127280
25	8376	0.935	89075	0.952	43514/132585
50	8351	0.944	89861	0.933	42938/135185
100	8301	0.933	88517	0.956	43259/135788
150	8251	0.944	87038	0.942	42403/134688
200	8201	0.914	85283	0.939	41699/135308
300	8101	0.916	85254	0.913	42456/137350
400	8001	0.938	86436	0.925	41390/138420
500	7901	0.917	83850	0.927	40502/137088
600	7801	0.928	82064	0.924	40433/136469

Table 6 extends the previous experiment by increasing the initial number of unbatched observations, n_0, by a factor of three from 8400 to 25200. This allows for larger n/m ratios than the previous experiment along with relatively large batch sizes. The desired probability is met by this increase in the number of initial unbatched observations at the cost of added computational expense. This experiment shows the validity and computational requirements of using R+ and KN+ RS methods on underlying processes that are not *iid* normal as long as batching methods are applied appropriately.

Table 6: R+ and KN+ Comparison Using Overlapping Batch Means while Varying Batch Size, $n_0 = 25200$

$\delta^* = 0.007878$, $P^* = 0.95$, $n_0 = 25200$, **LF**, $AR(1)$, $\phi = 0.22$					
		R+		**KN+**	
m	b_0	$\hat{P}(CS)$	$\hat{\bar{T}}$	$\hat{P}(CS)$	$\hat{\bar{T}}$
100	25101	0.982	272346	0.968	133642
200	25001	0.971	270824	0.940	130027
300	24901	0.969	273142	0.952	131120
400	24801	0.953	270408	0.957	135667
500	24701	0.968	269023	0.945	130849

Batch Method Summary

Data batching methods like BM and OBM allow for normal observations under certain conditions from simulations whose data fit a variety of distributions. BM requires a relatively large number of unbatched observations along with a sufficiently large batch size to ensure proper ranking and selection method performance. The OBM method requires fewer raw observations than the BM method for the same number of batched observations; however, OBM batches are correlated. This necessitates a large n/m ratio along with a large batch size m to ensure consistent variance estimation. Once asymptotic variance convergence is obtained, RS methods such as R+ and KN+ can meet the requirements of $P(CS)$.

4.3.4 Summary

Implementation of the Rinott (R+) and the Kim and Nelson (KN+) RS methods provides a baseline for the new RS methods developed in the next section. Several insights from these results also highlight potential improvements. First, embedded data estimators can potentially decrease the required number of raw observations. Second, the reliance of these methods on the initial number of observations motivates new methods whose total required observations are not sensitive to high or low initial observation settings. Finally, batching methods allow these RS methods to be used with simulated processes generating correlated output, but warrant a mechanism for confirming asymptotic variance convergence to ensure that the desired $P(CS)$ is achieved.

4.4 Ranking and Selection Method Development

There are several possible improvements to current RS methods. Note R+ and KN+ both rely on variance estimators from first-stage observed data to estimate the number of observations required in second and later stages. The first new RS method developed here uses variance estimators from current data, which is enabled by the data encapsulation methods discussed in section 4.2.2. Since the desired probability is often greatly exceeded, a second RS method introduces a reduction coefficient in the calculation determining the number of required raw observations. The third new RS method incorporates designer intuition about simulated system configuration performance. The fourth new method presented here uses embedded data calculations for the degrees of freedom to reduce the number of required raw observations. Lastly, a new method is explored which incorporates the current number of simulated system configurations still in contention for selection as the “best”. The remainder of this section explores corresponding extensions to current RS methods.

4.4.1 BGP Technique 1

Recall the KN+ method described in section 2.6.2. In this method the variance of the difference between observations from competing system configurations, $v_i^2 + v_l^2 = V_{il}^2$, is used to eliminate system configurations from analysis using:

$$W_{il}(r) = \max\left\{0, \frac{\delta}{2r}\left(\frac{h^2 m_0 V_{il}^2}{\delta^2} - r\right)\right\} \tag{30}$$

Note this variance estimator, V_{il}^2, is based on first-stage observations. The Benson/Goldsman/Pritchett (BGP) 1 method modifies the KN+ method by using the current variance estimator of the difference as enabled by the embedded data encapsulation methods presented in section 4.2.2.

The next experiment uses the LF configuration of means, a desired probability of $P^* = 1 - \alpha = 0.95$, $k = 6$ competing system configurations, and an indifference-zone parameter of $\delta^* = 0.30619$, and a batch size $m = 1$. The underlying simulated process for this experiment is $iid - N(\mu, \sigma)$ where $\sigma = 1.5$. 1000 independent replications were made for each experiment. The first-stage number of initial observations, n_0, varies in this experiment. Overall performance of the method, described by estimators of the probability of correct selection, $\hat{P}(CS)$, average number of required raw observations $\hat{\bar{T}}$, and the upper bound on the number of required raw observations, is shown below in Table 7. Recall the upper bound on the number of required raw observations is calculated during the second-stage of the KN+ method.

Table 7: BGP1 and KN+ Comparison Varying Initial Number of Observations

$\delta^* = 0.30619$, $P^* = 0.95$, **LF**, $iid - N(\mu,\sigma)$, $m = 1$				
	BGP1		**KN+**	
	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
$n_0 = 8$	0.989	234/945	0.990	277/928
$n_0 = 10$	0.982	207/760	0.968	235/764
$n_0 = 20$	0.975	159/543	0.968	162/533
$n_0 = 30$	0.985	147/478	0.971	152/481
$n_0 = 40$	0.986	142/469	0.992	149/468
$n_0 = 50$	0.984	140/444	0.963	141/446
$n_0 = 60$	0.959	139/444	0.957	137/420
$n_0 = 70$	0.965	138/434	0.975	140/432
$n_0 = 80$	0.967	139/427	0.967	141/428
$n_0 = 90$	0.965	140/422	0.972	143/422
$n_0 = 100$	0.971	141/420	0.968	143/419

Recall the assumption of strong consistency, where $\overline{X}_i(r) \to \mu_i$ as $r \to \infty$ with probability one. Here, μ_i is the steady-state mean from system *i* and $\overline{X}_i(r)$ is the sample mean based on *r* observations from system *i*. Assuming strong consistency, updated variance estimators are less than or equal to first-stage variance estimators guaranteeing BGP1 will perform at least as well, if not better, than the KN+ method. Observed performance of BGP1, in terms of the required number of raw observations, is marginally better than KN+, especially when the number of initial observations is small. $\hat{P}(CS)$ is not statistically differentiable between the methods. Note $\hat{P}(CS)$ meets or exceeds the desired probability in all conditions. BGP1 performance in terms of the average number of required raw observations, $\hat{\bar{T}}$, is used for comparative analysis for subsequent methods.

4.4.2 BGP Technique 2

BGP2 adds to BGP1 a reduction coefficient, R_c, to decrease the number of required observations. To do so, this method recognizes that the achieved $P(CS)$ of BGP1 and other methods often exceeds the specified requirement. The reduction coefficient reduces the conservatism of the method. Specifically, this method uses the following relationship during the screening process:

$$W_{il}(r) = \max\left\{0, R_c \frac{\delta}{2r}\left(\frac{h^2 m_0 V_{il}^2}{\delta^2} - r\right)\right\} \tag{31}$$

The reduction coefficient effectively increases the elimination rate during the screening phase of the method. The following experiment sets $R_c = 0.80$, with results shown in Table 8.

Table 8: BGP2 and BGP1 Comparison Varying Initial Number of Observations

$\delta^* = 0.30619$, $P^* = 0.95$, **LF**, $iid - N(\mu,\sigma)$, $m = 1$				
	BGP2 $(R_c = 0.80)$		**BGP1**	
	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
$n_0 = 8$	0.992	202/952	0.989	234/945
$n_0 = 10$	0.980	178/752	0.982	207/760
$n_0 = 20$	0.942	135/534	0.975	159/543
$n_0 = 30$	0.973	123/483	0.985	147/478
$n_0 = 40$	0.949	121/473	0.986	142/469
$n_0 = 50$	0.940	119/449	0.984	140/444
$n_0 = 60$	0.931	119/439	0.959	139/444
$n_0 = 70$	0.936	119/431	0.965	138/434
$n_0 = 80$	0.939	121/415	0.967	139/427
$n_0 = 90$	0.957	127/417	0.965	140/422
$n_0 = 100$	0.952	132/414	0.971	141/420

Improvement in terms of reduced observations is evident. The relative performance improvement also decreases as the number of initial observations, n_0, becomes large. Observe the desired probability is not achieved with several settings of the initial number of observations, n_0. Other experiments where the reduction coefficient is set to $R_c = 0.60$ indicate a greatly increased failure rate in achieving the desired probability. On the other hand, experiments with a reduction coefficient set to $R_c = 0.90$ generally achieved the desired probability. Based on these factors, the introduction of a reduction coefficient may aid in the application of RS methods in some situations, but must be carefully checked to ensure the desired probability is being achieved.

4.4.3 BGP Technique 3

Another possible performance improvement adds designer intuition to BGP1. For example, a designer may have some intuition, as a ratio value inferring some relative strength of one system configuration relative to others. This intuition can be used to take more initial observations from the "believed" best system configuration. Therefore, BGP3 will:

1. Sample a system configuration identified by designer intuition for a scaled number of additional observations during the first stage of the RS method.
2. Retain the intuitively selected system configuration, i.e., keep it in contention, until termination of the experiment.

Table 9 presents BGP3 experimental results when the designer has "good" intuition, i.e, selects the "best" system configuration. Two ratio values are selected intuitively; specifically, multipliers of the initial number of observations are $2n_0$ and $5n_0$.

Table 9: BGP3 and BGP1 Comparison Varying Initial Number of Observations

$\delta^* = 0.30619$, $P^* = 0.95$, **LF – Good Intuition**, $iid - N(\mu,\sigma)$, $m = 1$						
	BGP3 $(2n_0)$		**BGP3** $(5n_0)$		**BGP1**	
	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
$n_0 = 8$	0.999	232/880	0.993	226/831	0.989	234/945
$n_0 = 10$	0.998	202/718	0.999	197/693	0.982	207/760
$n_0 = 20$	0.989	153/508	0.983	143/512	0.975	159/543
$n_0 = 30$	0.983	138/452	0.955	133/462	0.985	147/478
$n_0 = 40$	0.985	135/445	0.955	128/436	0.986	142/469
$n_0 = 50$	0.971	131/429	0.941	153/410	0.984	140/444
$n_0 = 60$	0.953	130/409	0.942	174/389	0.959	139/444
$n_0 = 70$	0.983	125/398	0.925	171/387	0.965	138/434
$n_0 = 80$	0.962	128/398	0.943	173/388	0.967	139/427

For a small number of initial observations, “good” intuition results in a high estimated $\hat{P}(CS)$ and a lower number of total required observations. Also, a higher ratio value achieves a lower number of required observations. Thus, relatively strong and “good” intuition implies using a higher multiplier when the number of initial observations is small.

However, a large multiplier combined with a large number of initial observations can result in a failure to achieve the desired probability. Recall BGP3 calculates an upper bound on the number of raw observations that is directly proportional to the value of the first-stage variance estimator. Failure to achieve the desired probability is attributed to a relatively small upper bound, i.e., small first-stage variance estimator, on the number of observations for the “best” system configuration.

Table 10 highlights BGP3 experimental results when both “good” and “poor” intuition is used by the designer. BGP3 achieves the desired probability in all conditions. Both “good” and “poor” intuition impact the required number of observations in a logical manner, e.g., “good” intuition results in fewer required observations and vice versa. If designer intuition is completely random, then using BGP3 results in a higher expected number of observations. Also, the use of “good” intuition results in a lower number of required observations when the number of initial observations is larger.

Table 10: BGP3 Intuition Comparison Varying Initial Number of Observations

$\delta^* = 0.30619$, $P^* = 0.95$, **LF – Intuition**, $iid - N(\mu,\sigma)$						
	BGP3 $(2n_0)$ **POOR**		**BGP3** $(2n_0)$ **GOOD**		**BGP1**	
	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
$n_0 = 8$	0.990	246/883	0.999	232/880	0.989	234/945
$n_0 = 10$	0.983	214/711	0.998	202/718	0.982	207/760
$n_0 = 20$	0.971	169/509	0.989	153/508	0.975	159/543
$n_0 = 30$	0.982	159/454	0.983	138/452	0.985	147/478
$n_0 = 40$	0.978	153/444	0.985	135/445	0.986	142/469
$n_0 = 50$	0.987	154/420	0.971	131/429	0.984	140/444

Unconditional retention of the intuitively selected “best” system configuration may be overly cautious. Therefore, the following experiment explores changing BGP3 to only use intuition to scale the initial number of observations while not retaining the intuitively selected “best” system configuration should it be found to be no longer competitive. Table 11 highlights experimental results. Again, the desired probability is achieved in all conditions. The modified BGP3 technique penalizes "poor" intuition to a lesser extent. While the use of completely random intuition results in a slightly higher number of expected observations, "good" intuition offers improved computational performance.

Table 11: BGP3 without Retention Varying Initial Number of Observations

$\delta^* = 0.30619$, $P^* = 0.95$, **LF – Intuition**, $iid - N(\mu,\sigma)$, $m = 1$						
	BGP3 $(2n_0)$ POOR		**BGP3 $(2n_0)$ GOOD**		**BGP1**	
	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
$n_0 = 8$	0.988	237/887	0.992	231/885	0.989	234/945
$n_0 = 10$	0.983	206/722	0.984	204/715	0.982	207/760
$n_0 = 20$	0.989	158/504	0.983	154/510	0.975	159/543
$n_0 = 30$	0.987	151/455	0.979	139/456	0.985	147/478
$n_0 = 40$	0.967	146/442	0.976	136/445	0.986	142/469
$n_0 = 50$	0.974	143/417	0.971	131/432	0.984	140/444

Thus, BGP3 is appropriate when practitioner intuition is reliable, but only provides improved performance when using a relatively small number of initial observations. However, without a priori knowledge on the relationship of the indifference-zone parameter to the underlying variance, the selection of the initial number of observations can be problematic. Manipulating the indifference-zone parameter should not be considered because it ought to be selected as the point where the designer would not differentiate between competing system configurations and thus should depend on other considerations.

4.4.4 BGP Technique 4

The theoretical bounding of updated variance estimators, V_{il}^2, being less than or equal to first-stage estimators, assuming variance consistency, implies ranking and selection methods using updated variance estimators will always exhibit equal or increased computational performance compared to methods such as KN+. Recall both R+ and KN+ use first-stage variance estimators for second and later-stage calculations. Also, recalculation of the test statistic, $h^2 = 2\eta d$, with the current degrees of freedom, d, results in a monotonically decreasing value for h^2. Embedded calculation of the following relationships using updated variance estimators:

$$W_{il}(r) = \max\left\{0, \frac{\delta}{2r}\left(\frac{h^2 m_0 V_{il}^2}{\delta^2} - r\right)\right\} \tag{32}$$

$$\eta = \frac{1}{2}\{[2(1-(1-\alpha)^{1/(k-1)}]^{-2/d} - 1\} \tag{33}$$

are generally smaller than first-stage calculations of the same relationships. BGP4 thus incorporates embedded data estimators enabling updates of η and $W_{il}(r)$ at each stage as a heuristic RS method.

4.4.4.1 BGP4 Initial Performance Assessment

Table 12 below compares performance of BGP4 and BGP1. Experiment parameterization includes the LF configuration of means, a desired probability of $P^* = 0.95$, an indifference-zone parameter of $\delta^* = 0.30619$, $k = 6$ competing system configurations, and a batch size $m = 1$. The underlying simulated process for this experiment is $iid - N(\mu, \sigma)$ where $\sigma = 1.5$. 1000 independent replications were made in each experimental condition. The first-stage number of initial observations, n_0, varies in this experiment. Overall performance of the method, described by estimators of the probability of correct selection, $\hat{P}(CS)$, average number of required raw observations $\hat{\bar{T}}$, and the upper bound on the number of required raw observations is shown below.

Table 12: BGP4 and BGP1 Comparison Varying Initial Number of Observations

$\delta^* = 0.30619$, $P^* = 0.95$, **LF**, $iid - N(\mu,\sigma)$, $m = 1$				
	BGP4		**BGP1**	
	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
$n_0 = 8$	0.966	133/897	0.989	234/945
$n_0 = 10$	0.976	132/757	0.982	207/760
$n_0 = 20$	0.985	133/541	0.975	159/543
$n_0 = 30$	0.982	130/476	0.985	147/478
$n_0 = 40$	0.981	130/466	0.986	142/469
$n_0 = 50$	0.984	133/440	0.984	140/444
$n_0 = 60$	0.989	133/424	0.959	139/444
$n_0 = 70$	0.988	133/419	0.965	138/434
$n_0 = 80$	0.979	135/416	0.967	139/427
$n_0 = 90$	0.975	135/417	0.965	140/422
$n_0 = 100$	0.964	142/416	0.971	141/420

Observe there is significant reduction in the number of required observations when the number of initial observations, n_0, is small. Additionally, the number of required observations for BGP4 is approximately flat when the number of initial observations is less than 100, implying the selection of n_0 has little effect on method performance for relatively small n_0 values. Figure 18, below, graphically confirms this observation.

Figure 18 compares the ranking and selection methods KN+, BGP1, and BGP4. KN+ and BGP1 have similar performance. Both methods produce $\hat{\bar{T}}$ values that are concave in nature, illustrated by the decreasing then increasing number of required observations as the number of initial observations, n_0, becomes larger. BGP1 outperforms KN+ to some extent when n_0 is small. On the other hand, BGP4 exhibits different behavior in that its number of total required observations is approximately constant when the number of initial observations is small. This is a result of using embedded variance estimators. Therefore, BGP4 is promising as the selection of the initial number of observations, n_0, has little, if any, effect on the number of total required observations for small n_0 values.

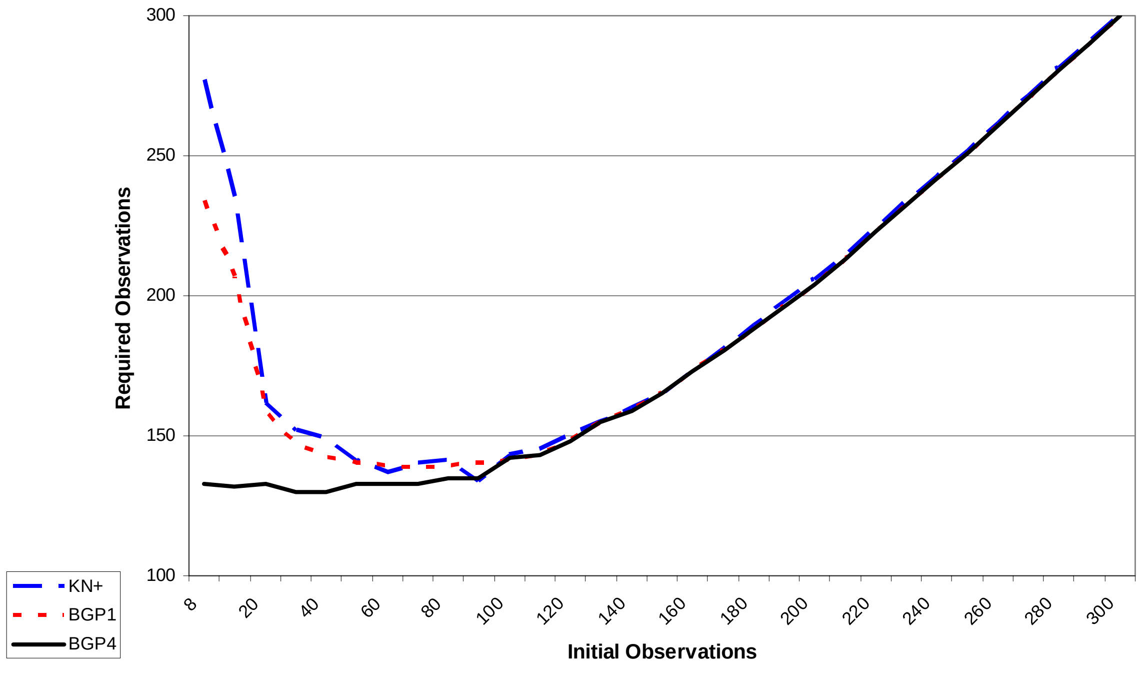


Figure 18: KN+, BGP1, and BGP4 Comparison of Initial Number of Observations versus Required Number of Observations

4.4.4.2 BGP4 AR(1) Batch Means Performance

Application of BGP4 to a simulation with an underlying autoregressive, $AR(1)$, process using Batch Means (BM) for observation acquisition ascertains the robustness of the method to the correlated output often generated by time-based simulations. Experiment parameterization involved setting $\phi = 0.22$, $n_0 = 4200$, $P^* = 0.95$, $k = 6$ competing system configurations, and $\delta^* = 0.019298 = \sqrt{1.564/4200}$ while varying both batch size, *m*, and the initial number of batched observations where $b_0 = n/m$. Results from this experiment are shown in Table 13 for simulated configurations possessing an underlying autoregressive process. Recall an $AR(1)$ process mimics systems with time-based observations or some type of correlation between the data.

Table 13: BGP4 and KN+ Comparison Using Batch Means while Varying Batch Size with Mildly Correlated Data

$\delta^* = 0.019298$, $P^* = 0.95$, $n_0 = 4200$, **LF**, $AR(1)$, $\phi = 0.22$					
		BGP4		**KN+**	
m	b_0	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
10	420	0.959	20550/65428	0.968	20912/64676
25	168	0.958	22199/69994	0.964	22834/69939
50	84	0.957	22313/73859	0.972	23005/73263
100	42	0.948	22785/82173	0.976	25219/82739
150	28	0.950	23010/90078	0.955	30191/94992
200	21	0.959	23423/96782	0.974	29919/99284
300	14	0.949	23617/109016	0.962	33371/110908

BGP4 achieves the desired probability, statistically, in all conditions. At a small number of initial batches, b_0, BGP4 significantly outperforms the KN+ method. This is attributed to the use of embedded data estimators enabling the method to anneal/conform to the underlying distributions. Note asymptotic variance convergence of the underlying process is a requirement for proper method performance in terms of achieving the desired probability.

Table 14 extends the previous experiment by changing $\phi = 0.5$ to increase the correlation within the simulated $AR(1)$ process. This increase in ϕ induces an underlying process with higher variability. The indifference-zone parameter is set to $\delta^* = \sqrt{3.0/4200} = 0.026726$.

Table 14: BGP4 Using Batch Means while Varying Batch Size with Moderately Correlated Data

$\delta^* = 0.026726$, $P^* = 0.95$, $n_0 = 4200$, **LF**, $AR(1)$, $\phi = 0.5$					
		BGP4 $(\phi = 0.5)$		**BGP4** $(\phi = 0.22)$	
m	b_0	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
100	42	0.947	22621/82101	0.948	22785/82173
150	28	0.946	22828/89693	0.950	23010/90078
200	21	0.958	23024/95294	0.959	23423/96782
300	14	0.949	23682/111031	0.949	23617/109016

BGP4 achieves the desired probability, statistically, in all conditions. Observe the slight reduction in $\hat{P}(CS)$ with the higher ϕ. This experiment indicates encouraging BGP4 performance on batched observations from a moderately correlated underlying process.

Generally, time based simulations can produce highly correlated output. For example, measuring separation between arriving entities to some location will naturally produce highly correlated data. To ascertain the robustness of BGP4 in a highly correlated environment, an $AR(1)$ process with $\phi = 0.95$ provides an appropriate test case. The indifference-zone parameter is set to $\delta^* = \sqrt{39.0/4200} = 0.096262$ for this experiment. Note Equal Spacing (ES) of means is assumed for the configurations under contention for selection as the “best”. This assumption gives us an example in which the competing system configurations are differentiated by some significant factor. We would expect to do well here, in terms of achieved $P(CS)$, since this is a “highly favorable” configuration of the means. Experimental results are shown in Table 15.

Table 15: BGP4 Using Batch Means while Varying Batch Size with Highly Correlated Data

$\delta^* = 0.096362$, $P^* = 0.95$, $n_0 = 4200$, **ES**, $AR(1)$, $\phi = 0.95$			
		BGP4	
m	b_0	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
100	42	0.961	10881/64898
200	21	0.981	12172/86007
300	14	0.970	13406/105144

BGP4 achieves the desired probability in all conditions. This experiment shows promise for the use of BGP4 with highly correlated output given an assumed ES configuration of means for the competing system configurations.

4.4.4.3 BGP4 AR(1) Overlapping Batch Mean Performance

Application of BGP4 to an autoregressive, $AR(1)$, process using Overlapping Batch Means (OBM) for observation acquisition further explores the applicability of the method. This experiment sets $\phi = 0.22$, $n_0 = 25200$, $P^* = 0.95$, $k = 6$ competing system configurations, and $\delta^* = 0.007878 = \sqrt{1.564/25200}$, while varying both batch size, *m*, and the initial number of batched observations where $b_0 = n - m + 1$. Note the difference between OBM and BM where batched observations are obtained from the relationship $b = n/m$. Results are shown in Table 16.

Table 16: BGP4 and KN+ Comparison Using Overlapping Batch Means while Varying Batch Size

$\delta^* = 0.007878$, $P^* = 0.95$, $n_0 = 25200$, **LF**, $AR(1)$, $\phi = 0.22$					
		BGP4		**KN+**	
m	b_0	$\hat{P}(CS)$	$\hat{\bar{T}}$	$\hat{P}(CS)$	$\hat{\bar{T}}$
100	25101	0.960	133781	0.968	133642
200	25001	0.959	132874	0.940	130027
300	24901	0.961	133045	0.952	131120
400	24801	0.967	135944	0.957	135667
500	24701	0.972	134810	0.945	130849

BGP4 achieves the desired probability in all conditions. Computational reduction relative to KN+ is not evident in this experiment. This is attributed to the selection of δ^* and n_0 where KN+ achieves efficient performance. The relevant aspect of this experiment is BGP4 achieving the desired probability when using OBM for observation acquisition.

4.4.4.4 BGP4 Summary

BGP4 is a new ranking and selection method. BGP4 anneals or conforms in some sense to the underlying simulated process. Clearly, BGP4 offers increased performance in several areas. Given asymptotic variance convergence, BGP4 performs as well or better than other RS methods in terms of computational requirements while achieving the desired probability. Additionally, BGP4 avoids “guessing” on the initial number of observations. Rather, asymptotic variance convergence and normally distributed observations are the only requirements for proper method performance. Note that batched $AR(1)$ observations are marginally normal. A formal statement of the BGP4 method follows:

For two systems *i* and *l*, the asymptotic variance of the difference between the two systems, $v_i^2 + v_l^2$, is estimated by applying one of the estimators presented in section 2.5 on the differenced series $D_{ilj} = X_{ij} - X_{lj}$, $j = 1,2 \ldots,$.

Setup: Select confidence level $1-\alpha$, indifference-zone parameter $\delta > 0$, first-stage sample size $n_0 \geq 2$, and batch size $m < n_0$. Selection of n_0 must ensure asymptotic variance convergence. Calculate

$$\eta = \frac{1}{2}\{[2(1-(1-\alpha)^{1/(k-1)}]^{-2/d} - 1\} \tag{34}$$

Initialization: Let $I = \{1,2,\ldots,k\}$ be the set of systems still in contention, and let $h^2 = 2\eta d$. Obtain n_0 observations X_{ij}, $j = 1,2,\ldots,n_0$, from each system $i = 1,2,\ldots,k$.

For all $i \neq l$ compute $m_0 V_{il}^2$, the sample asymptotic variance of the difference of systems *i* and *l*.

Let

$$N_i = \left\lfloor \frac{h^2 m V_{il}^2}{\delta^2} \right\rfloor \tag{35}$$

and

$$N_i = \max_{l \neq i} N_{il} \tag{36}$$

Here $N_i + 1$ is the maximum number of observations that can be taken from system *i*. If $n_0 \geq \max_i N_i + 1$ then stop and select the system with the largest $\overline{X}_i(n_0)$ as the best. Otherwise, set the observation counter $r = n_0$ and go to Screening.

Screening: Set $I^{old} = I$.

update with the current degrees of freedom, d:

$$\eta = \frac{1}{2}\{[2(1-(1-\alpha)^{1/(k-1)}]^{-2/d} - 1\} \tag{37}$$

update with the current degrees of freedom, d, and η:

$$h^2 = 2\eta d \tag{38}$$

update with the current test statistic, h^2, and estimated variance, V_{il}^2:

$$W_{il}(r) = \max\left\{0, \frac{\delta}{2r}\left(\frac{h^2 m V_{il}^2}{\delta^2} - r\right)\right\} \tag{39}$$

Let

$$\begin{aligned} I = \{i : i \in I^{old} \text{ and } \overline{X}_i(r) \geq \overline{X}_l(r) - W_{il}(r), \\ \text{for all } l \in I^{old}, l \neq i\} \end{aligned} \tag{40}$$

Stopping Rule: If $|I| = 1$, then stop and select the system whose index is in *I* as the best. Otherwise, take one additional observation $X_{i,r+1}$ from each system $i \in I$ and set $r = r + 1$. If $r = \max_i N_i + 1$, then stop and select the system whose index is in *I* and has the largest $\overline{X}_i(r)$ as the best. Otherwise, repeat the screening process.

BGP4 outperforms other ranking and selection methods, such as R+ and KN+, when the underlying simulated process is either $iid - N(\mu, \sigma)$ or $AR(1)$. Performance of BGP4 with BM and OBM batching methods on correlated data from an $AR(1)$ process demonstrates applicability for different variance estimators.

4.4.5 BGP Technique 5

One possible enhancement to BGP4 involves the use of embedded estimators to update the relationship:

$$\eta = \frac{1}{2}\{[2(1-(1-\alpha)^{1/(k-1)}]^{-2/d} - 1\} \tag{41}$$

by the number of system configurations, *k*, still in contention versus the number of total initial system configurations. Recall η is an intrinsic component of h^2 used in the screening phase of the KN+ method discussed in section 2.6.2.

Using the same experimental parameterization as the BGP4 experiment, Table 17 compares performance of BGP5 with the BGP4 method. The use of embedded estimators to update η in this manner is ineffective; even though the number of required observations has decreased, BGP5 fails to achieve the desired probability.

Table 17: BGP5 and BGP4 Comparison Varying Initial Number of Observations

$\delta^* = 0.30619$, $P^* = 0.95$, **LF**, $iid - N(\mu,\sigma)$, $m = 1$				
	BGP5		**BGP4**	
	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound	$\hat{P}(CS)$	$\hat{\bar{T}}$ /Upper Bound
$n_0 = 8$	0.852	109/897	0.966	133/897
$n_0 = 10$	0.887	109/760	0.976	132/757
$n_0 = 20$	0.875	106/545	0.985	133/541
$n_0 = 30$	0.886	107/480	0.982	130/476
$n_0 = 40$	0.895	104/469	0.981	130/466
$n_0 = 50$	0.856	108/439	0.984	133/440

4.4.6 Method Development Summary

Incorporation of embedded variance estimators extends current ranking and selection methods by achieving the desired probability while decreasing computational requirements. BGP1 offers better computational performance over current methods, such as KN+, when the number of initial observations is small. However, when the number of initial observations is large there is little, if any, improvement. The latter case is equivalent to the experimenter obtaining more observations in the initial stage than is required for the experiment.

Current RS methods often exceed the desired probability. Introduction of a reduction coefficient can result in achieving the desired probability while increasing computational efficiency. However, arbitrary selection of the reduction coefficient can result in failure to achieve the desired probability. In fact, the practitioner can only select a reduction coefficient with a priori knowledge of the underlying system. Hence, BGP2 should only be used in strictly defined experimental environments as this method lacks theoretical rigor.

Use of designer intuition resulted in better computational performance under certain conditions. When the number of initial observations is small, strong and accurate intuition decreases the required number of observations. However, random or “poor” intuition degrades RS method performance. Of note, BGP3 incorporates designer subjectivity into the experimental process.

BGP4 offers significantly increased computational efficiency compared to other RS methods. Use of embedded data estimators enables this method to “anneal” itself to the underlying processes in contention. Selection of the initial number of observations needs only ensure asymptotic variance convergence for this method to perform properly. Observations are assumed to be normally distributed but can be obtained from batching methods. Known RS methods such as R+ and KN+ have the same requirements. However, the lack of the need to "guess" the number of initial observations differentiates BGP4 from other RS methods.

BGP4 is a new approach to RS methods. This method avoids the pitfalls of reliance on the initial number observations, n_0. Rather, BGP4 incorporates embedded estimators to enable a form of annealing to the underlying process. The use of current estimators enables tight control of the process. Application of this technique to simulated configurations with an underlying $AR(1)$ process highlights the robustness of the method.

BGP5 uses the number of competing configurations still in contention for selection as the "best" during the screening process of the method. This RS method fails to achieve the probability requirement. Here, the use of embedded estimators is inappropriate.

4.5 Ranking and Selection Method Summary

Adaptive control techniques, such as ranking and selection, enable differentiation between competing simulated system configurations. BGP4 is a new method that outperforms known ranking and selection methods in terms of computational efficiency. Increased performance is obtained by incorporation of embedded data estimators. BGP4 relies on the same assumptions as R+ and KN+. Specifically, observation normality, variance consistency, and an underlying stationary process are assumed. In addition, batching methods allow transformation of correlated data into normal observations under certain conditions.

CHAPTER 5

TEST CASE: NATIONAL AIRSPACE SYSTEM ANALYSIS

Parallel and Distributed Simulation (PDS) techniques along with Ranking and Selection (RS) methods enable analytic comparison of large-scale simulated system configurations of a real-world process. While previous chapters presented PDS and RS methods in a controlled environment for testing and evaluation, this chapter highlights the application of these methods to an existing simulation of a complex system. Specifically, PDS and RS methods are applied to the Reconfigurable Flight Simulator (RFS), an existing large-scale hybrid simulation, to assess aircraft separation with differing arrival route densities in the National Airspace System (NAS). Additionally, diagnostics for appropriate simulation parameterization are presented.

5.1 Air Traffic Simulation

One example of a complex system is the National Airspace System (NAS). Within the NAS, Air Traffic Control (ATC) systems ensure the safe travel of an aircraft from one airport to another while Air Traffic Management (ATM) systems schedule and sequence aircraft to increase throughput and reduce delay. Prior to departure the flight crew is given routing information from both automated and human components of the system. This route is developed accounting for regulations, expected weather conditions, and traffic density. During departure, commercial aircraft follow specific directions on speed, heading, and altitude on a path that includes navigational points called “fixes”. En route, the aircraft will traverse one or more flight sectors that are managed by Air Route Traffic Control Centers (ARTCC) manned by human controllers assisted by a variety of aids. During arrival into a major airport, commercial aircraft generally follow a published procedure called a Standard Terminal Arrival Route (STAR) until the final approach.

This section provides a general description of ATC and ATM simulations. Models of ATC/ATM systems can be composed of human performance parameters, equipment characteristics, and regulatory procedures. Simulation of these models generally looks exclusively at factors such as capacity or safety. Capacity is often measured in throughput, or entities per time unit, that accomplish an activity, such as a plane arriving at a gate. Beyond a measure of performance, capacity directly relates to profit. Also, increased capacity is needed to meet anticipated future demand. On the other hand, safety is usually a discrete count of entities that violate specific criteria, for example a minimum separation distance between aircraft.

ATM simulation can determine the impact of flight restrictions on delay, throughput, and traffic congestion. Wieland (1998) describes the Detailed Policy Assessment Tool (DPAT) as a large-scale simulation capable of calculating traffic conditions for entire airspace regions, for example the continental United States. DPAT models the NAS as a sequence of capacitated resources in a parallel and discrete-event manner. The parameters used within DPAT are obtained from external models. DPAT has successfully simulated NAS operations for the entire continental United States faster than real-time for specific models.

The Total Airspace and Airport Model (TAAM) simulation is a high-fidelity simulation modeling NAS components such as gates, terminals, taxiways, and airspace. As one example, Holden and Wieland (2003) incorporated simulation optimization methods with TAAM to optimize runway scheduling. For this particular analysis, the scheduling impact of adding a new runway was simulated. Potentially, this method could also assist controllers with the allocation of aircraft to runways.

ATM simulation can also provide predictive insight on the impact of new equipment on airport throughput. For example, Schwartz et al. (1997) describe the use of simulation to evaluate the introduction of new Flight Management System (FMS) equipment in aircraft cockpits along with new routing procedures. They assumed that more sophisticated, but higher cost, FMS equipment corresponded to decreased controller-pilot verbal communication. Then, they simulated various combinations of traffic throughput and percentage of FMS equipped aircraft. Note that the capability of installed FMS equipment also varied in terms of acquisition cost. This method of sensitivity analysis provided insight not only that capacity could be increased by equipment fielding but it offered a cost-benefit element for determining the required sophistication in new FMS equipment.

Simulation of aircraft routing procedures has also been pursued as a method to increase capacity. Tofukuji (1993) provides an example in which various routing configurations were simulated to assess throughput. Results from this experiment included a relationship between throughput and required controller interventions. Additionally, this experiment compared existing route configurations along with proposed modifications.

Simultaneous impacts of changes on both capacity and safety have also been investigated through the use of simulation. For example, Zeghal and Hoffman (2000) explored model performance of ATC operations where the requirement of maintaining separation was delegated to individual aircraft. Here the sequencing of self-separating aircraft was simulated to predict future capacity and controller workload. Safety, in this case violation of a minimum separation threshold, was indirectly assessed using rules for sequencing aircraft that ensured safe separation.

Increasing use of simulation as a design and analysis activity implies larger and more complex simulations. Combination of discrete-event and continuous-time models into hybrid simulations will complicate metric analysis. However, this combination is necessary to provide realistic representation of complex systems such as the National Airspace System.

Modeled ATC and ATM systems have been simulated in an effort to obtain predictive measures of performance by numerous agencies with varying fidelity. Common to all efforts is the need for metric assessment and computational efficiency. Application of adaptive control techniques within a distributed simulation architecture not only reduces the computational requirement but speeds experimental execution. Versatile, embedded data encapsulation methods enable these control techniques.

5.2 Reconfigurable Flight Simulator (RFS)

The Reconfigurable Flight Simulator is used as a test case for several reasons. First, it is hybrid simulation modeling a complex system that cannot be simplified for an analytic solution without loss of fidelity. Second, it is a significant development in terms of personnel-hours as well as high-level software engineering. Minor modifications within the RFS software architecture, presented later, bode well for simulating other existing complex systems. Also, as the name implies, RFS is easily initialized for alternative configurations of the NAS by the use of formatted text configuration files. Lastly, RFS supports analysis of both discrete and continuous state variables.

Pritchett and Ippolito (2000) discuss the Object-Oriented (OO) structure and capabilities of the RFS. Also, Lee, Pritchett, and Goldsman (2001) detail the RFS timing mechanisms and their application to a hybrid, agent-based simulation of the National Airspace System. The OO structure of RFS is extensible and modular. Instantiation of the base classes produces objects that compose the simulation; these objects can be configured by a script file during initialization. In this context, an object is also considered an agent if it can autonomously interact with other agents while pursuing a particular goal or set of goals. Note that each agent is also self-describing in terms of identity, performance parameters, and current state. Combined agent behavior models complex system performance. Other objects in the simulation may not have two-way interactions with the agents, but instead serve other purposes such as graphic displays, date loggers, and analyzers.

The RFS architecture is shown in Figure 19. The simulation object is the overall controller of the simulation and manages all callback messages to other components. The timer object maintains the temporal state of the simulator and facilitates both continuous and discrete agent update timing mechanisms. Arrows in this diagram correspond to communication between agents. Lists within the architecture track corresponding agents in the simulation. Agents are included in the simulation by calling dynamic link libraries (.dll) enabling both modular development and rapid reconfiguration. Configuration of the simulation during initialization and runtime is accomplished through the use of pseudo-code and formatted text configuration files.

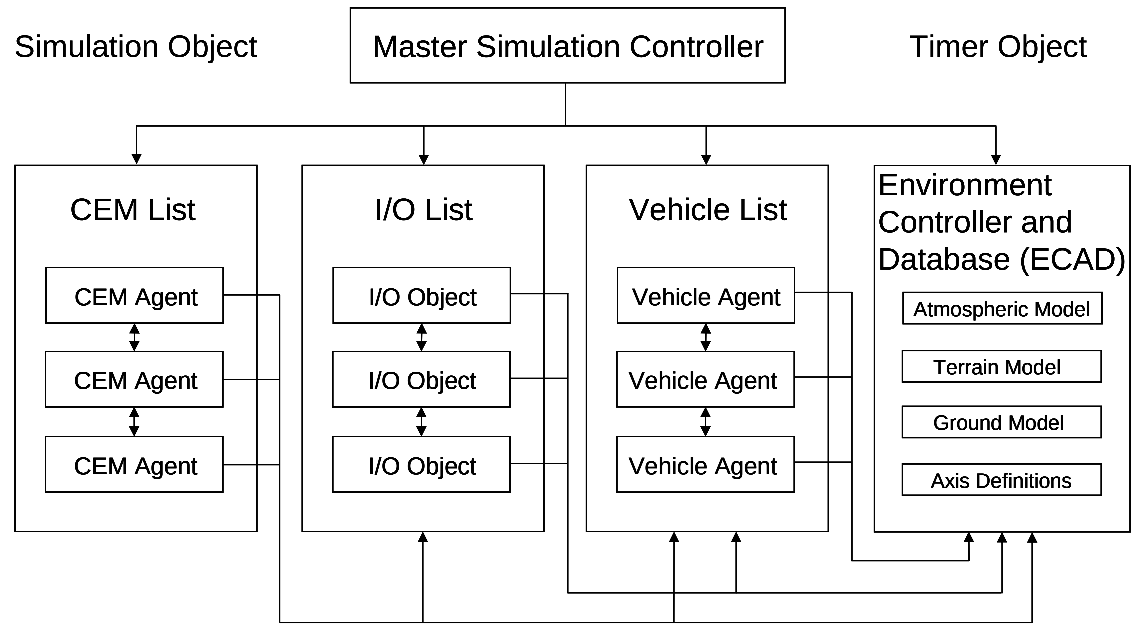


Figure 19: Reconfigurable Flight Simulator Architecture

The Environment Controller and Database (ECAD) object maintains the simulation environment. Environmental effects such as wind, terrain, and axis systems can be loaded as needed to ensure that the simulation environment is coherent.

Input/Output objects (I/O) provide mechanisms for both graphical and textual manipulation. For example, graphical output can take the form of an ATC display or a view of vehicle instrumentation. Text output is supported in ASCII format. Additionally, I/O objects support communications with other simulations and hardware. The I/O list manages all I/O objects.

Through inheritance of the base vehicle class, vehicle agents may be modified to represent continuous-time models of aircraft, ground vehicles, etc. in arbitrary numbers. This base class provides interfaces for communication to other vehicles and the simulation object. State variables, such as position, are available to these interfaces. Additionally, the simulation object can relay elements of the simulation status via these interfaces and vehicles can access the ECAD and I/O objects. All vehicles are maintained on the vehicle list to facilitate management and control. Several vehicles have been developed, such as a waypoint following aircraft, which can be used for complex system analysis.

Controller, Event, and Measurement (CEM) agents have access to the vehicle list, input/output list, and the ECAD object. CEM objects are typically extensions to a base class to complete a particular task. For example, the Measurement Management Agent (MMA) is a CEM agent that measures relative differences between or pairing interaction of agents in the simulation. One use of this agent to date is for adjusting vehicle update times to separate agents (aircraft), in air traffic control, to prevent collisions (Lee, 2002).

Access to base classes within the RFS is generally accomplished through the use of pointers and standardized interfaces. Object Data/Method Extensions (ODME) provide an alternative for invoking function calls or accessing data in RFS objects. Basically, ODME allows for extension of existing interfaces by allowing objects to specify data and methods available to other objects. Note that ODME allows for different objects to pass data without sharing header files.

The inherent modularity of RFS simplifies incorporation of ranking and selection methods. Here, the modularity allows for easy integration of new modules into the simulation. Additionally, existing RFS modules are extensible in nature. This allows for the minor modifications needed by the adaptive control structure. The following section details RFS module extension and new module development.

5.3 Reconfigurable Flight Simulator Module Development

The Reconfigurable Flight Simulator provides a modular, extensible, and reconfigurable architecture for use in the analysis of a complex system. This section highlights minor modifications to existing modules and new module development. Together, these modules enable adaptive control of RFS within a distributed simulation environment. Note the functionality of these modules can be generalized for the integration of adaptive control and PDS techniques to any existing large-scale simulation.

5.3.1 Simulation Controller (SC)

The Simulation Controller (SC) is a CEM agent that enables external control of the RFS. External control includes simulator commands of “PAUSE”, “UNPAUSE’, and “TERMINATE”. The “PAUSE” and “UNPAUSE’ commands allow the adaptive controller to command sufficient observation acquisition from each competing configuration during ranking and selection. The “TERMINATE” command is used when a configuration is no longer in contention for selection as the “best”. The SC also broadcasts an “EXIT” status after successful simulation termination.

5.3.2 Data Analyzer (DA)

The Data Analyzer (DA) is a CEM agent that monitors and calculates both mean and variance estimators for specified ODME variables. Figure 20 shows the general structure of this object. Flexible implementation of the DA created the ability for data encapsulation of single or grouped objects. For example, a DA can encapsulate data from a single vehicle or from all vehicles of a specific type. Additionally, the DA allows the practitioner to define logical data clusters, i.e. group variables. In this example, the data group POSITION clusters variables that include latitude, longitude, and altitude. The Interval Sampler (IS) is a CEM agent that allows for dynamic runtime setting of sampling methods. The sampling method may be synchronous or asynchronous with a specified time step and associated overlap. The overlap allows for obtaining observation data from agents possessing an update time within a certain boundary of the current sampling time.

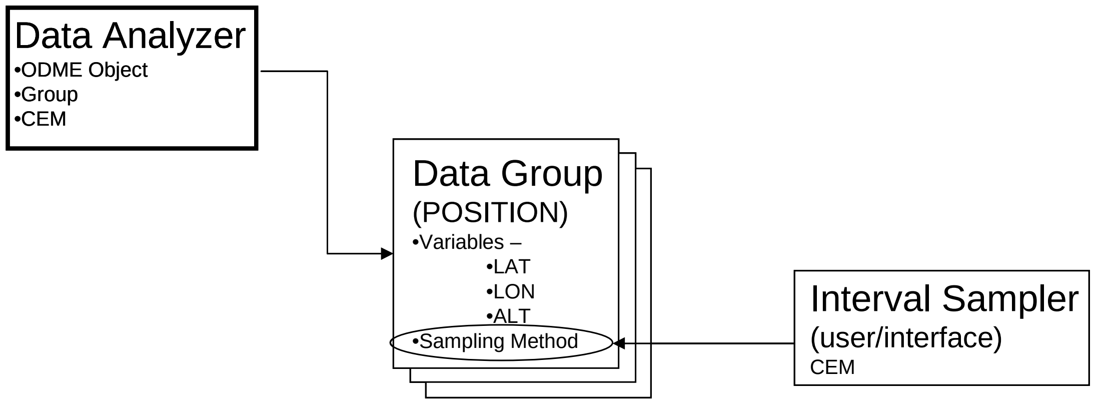


Figure 20: Data Analyzer Object Example Where it is Capturing Aircraft Position

A unique facet of the DA is embedded calculation of mean and variance estimators, precluding storage of historical data beyond summed and summed squared values for X_i as shown in section 4.2.2. Note the absence of variable under/overflow ensures estimator accuracy. Hard disk storage of these estimators in runtime specified locations enables external monitoring.

The computational overhead from using embedded estimators of this sort was assessed by running the same simulated configuration without embedded statistical analysis (NOSTAT), with embedded statistical analysis (STAT), and lastly with both the embedded statistical analysis and the distributed simulation client module (RFS Client). Figure 21 below highlights the overall results. Addition of embedded statistical analysis increased the computational expense of obtaining a specified number of observations by less than 1% in this example. Here, the RFS client module increased the overall expense by less than 2% for the same number of observations. In practical terms, arrivals for an operational day at Atlanta International Airport can be simulated on a single dual processor 2.2 GHz workstation with 512 megabytes of RAM in approximately 160 computer-minutes. An additional 3 minutes of workstation time allows for embedded statistical analysis in this example. Note the computational expense of embedded statistical analysis is inversely proportional to the expense of running the simulation. The impact of embedding statistical analysis is small when the computational requirements of the simulated configuration are large and vice versa.

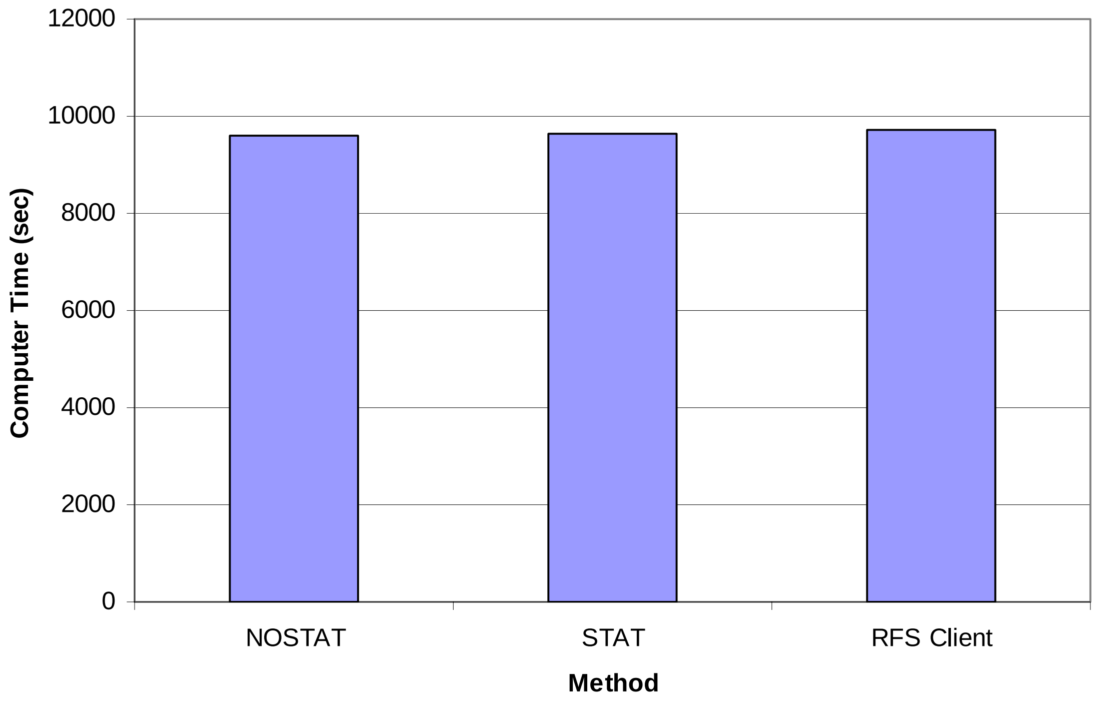


Figure 21: Software Infrastructure Overhead Comparison

For analyzing these results, note that users of an existing simulation generally record all data from a simulation for follow on analysis. The computational cost of this output and storage is not analyzed here. Presumably, however, it can represent both an increase in runtime and a subsequent analysis process that is necessary only in the "NOSTAT" condition.

5.3.3 Measurement Management Agent (MMA)

The previously established Measurement Management Agent (MMA) was a CEM object that measured relative differences between or interaction of pairs of agents in the simulation. In this test case, the paired objects are two individual aircraft, and extension of the MMA involved calculation of the average minimum, mean, and maximum distance between them. These calculations are available as ODME variables to other simulation modules.

5.4 Example NAS Scenario: Arrivals on Macey Two STAR to ATL

An Air Traffic Control (ATC) scenario with varied configurations provides an interesting large-scale simulation as a test case for the application of adaptive control and distributed simulation techniques. Specifically, different arrival routing density configurations for the Atlanta International Airport (ATL) Macey Two Standard Terminal and Arrival Routing (STAR) procedure are compared. Figure 22, below, highlights the Macey Two STAR. Of interest, the intersection at MACEY involves the incorporation or merging of traffic from the navigation aids Volunteer (VXV) and Spartanburg (SPA) and the "fix" AVERY. Arriving aircraft are assigned to one of these three paths by an air traffic manager much earlier in the flight depending upon the direction of arrival and expected aircraft density on each path. Once on a path, an air traffic controller maintains spacing between the aircraft.

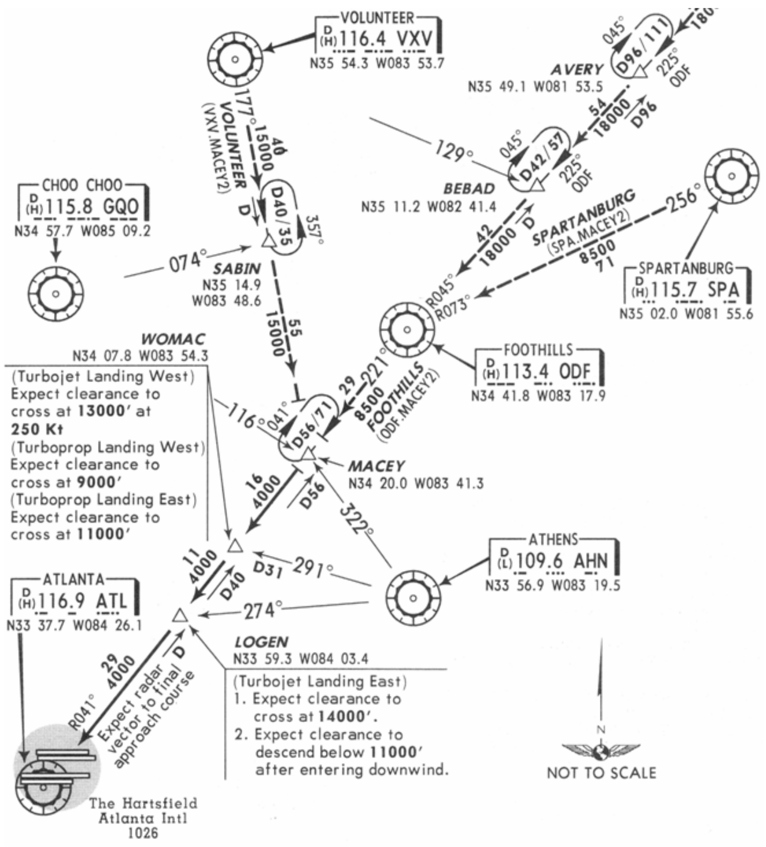


Figure 22: Atlanta International Macey Two Arrival STAR

RFS waypoint following aircraft (WPT) agents model arriving aircraft for this scenario. Each WPT agent uses numerical integration routines to update state variables including speed, heading, latitude, longitude, and altitude. The trajectory of WPT agents is defined by a list of waypoints initialized at instantiation. In this test case, WPT agents adjust their internal dynamics to cross each waypoint at a specified speed.

WPT agents are instantiated by the RFS Random Plane Generator (RPG) agent with initial performance parameters. The RPG agent creates WPT agents based on a random stationary Poisson process. The inter-arrival time for this Poisson process is set at RFS initialization. While actual arrivals to ATL are more closely modeled by a nonhomogeneous Poisson process, this simplification still allows for relevant system analysis. Note that generated WPT agents are added to the simulation vehicle list.

The RFS ATC agent models the air traffic controller. The ATC agent monitors waypoint following aircraft agents to ensure safe separation. The ATC agent maintains a list of WPT agents within a defined sector and provides calculated speed and heading commands to the WPT agents. The ATC agent also determines WPT agent sequencing in merging arrival streams. Additionally, the ATC agent models missed communication, communication delay, and misinterpreted command behavior (Lee, 2002).

Currently, approximately 615 aircraft arrive daily at ATL. The majority of these aircraft arrive between 6 am and 12 pm. This equates to an approximate 100 second inter-arrival time between aircraft, although this can be much higher during banks of arriving aircraft. Varying the allocation of aircraft on the three merging paths of the Macey Two STAR approach provides comparable configurations for this test case.

Table 18 presents the three route density configurations under analysis. Recall arriving aircraft merge from the navigation aids Volunteer (VXV) and Spartanburg (SPA) and the “fix” AVERY on the Macey Two STAR. Configuration C1 is the base case with equal 300 second expected inter-arrival times on each of the three arrival paths for a system-wide expected inter-arrival time of 100 seconds. Configuration C2 involves a higher arrival density, i.e., a lower inter-arrival time, on the northern path resulting in a system-wide expected inter-arrival time of approximately 83 seconds. Lastly, configuration C3 involves higher arrival densities on the two southern paths with the same system-wide expected inter-arrival time as configuration C2.

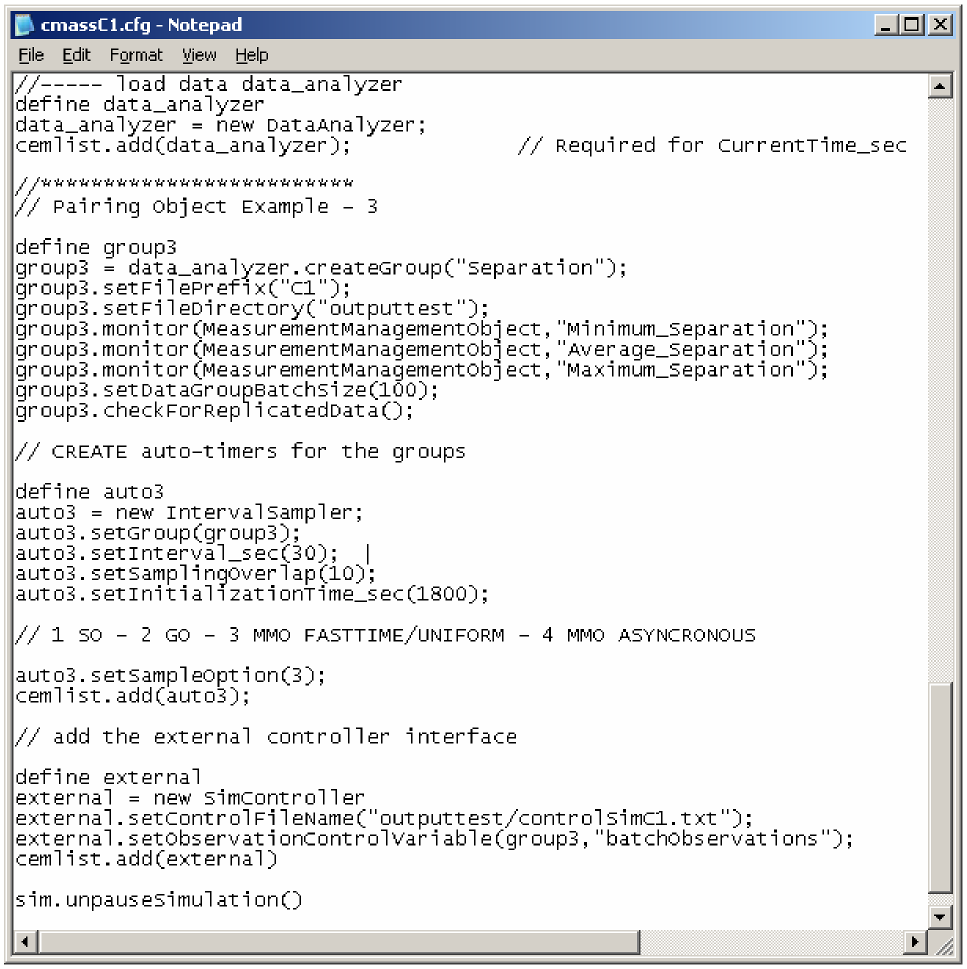
Table 18: Test Case Configuration Descriptions

Configuration	Expected Inter-Arrival Time (sec)
C1	VXV – 300 AVERY – 300 SPA – 300
C2	VXV – 150 AVERY – 375 SPA – 375
C3	VXV – 500 AVERY – 200 SPA – 200

Varying route densities in this manner addresses questions about efficient and safe allocation of aircraft to arrival paths. While central ATM seeks to control the overall number of aircraft arriving into ATL, traffic problems induced by such factors as weather can demand redistribution of these aircraft onto the arrival paths.

One metric of performance is the average minimum separation distance between aircraft. A larger value for this metric is considered safer. Without having to model all the factors contributing to a near-miss or aircraft collision (NMAC) event, low average minimum aircraft separation is a sufficient condition for such safety problems. It can also be inferred that a smaller average minimum separation implies a reduction in allowable reaction time from both pilots and controllers. The indifference-zone parameter, δ^*, is set to 1500 feet for this metric. This equates approximately to a six-second reaction time differential for pilots and controllers.

Initializing the RFS for this scenario is accomplished through the use of formatted text configuration files. Note all RFS modules are initialized in a similar fashion. A sample configuration file is shown in Figure 23. Note the commands to set simulation parameters. ODME variables, such as minimum_separation, for a MMA will be monitored by a DA. Lastly, external control files for the simulation are specified.



```
//----- load data data_analyzer
define data_analyzer
data_analyzer = new DataAnalyzer;
cemlist.add(data_analyzer);              // Required for CurrentTime_sec

//*****************************
// Pairing Object Example - 3

define group3
group3 = data_analyzer.createGroup("Separation");
group3.setFilePrefix("C1");
group3.setFileDirectory("outputtest");
group3.monitor(MeasurementManagementObject,"Minimum_Separation");
group3.monitor(MeasurementManagementObject,"Average_Separation");
group3.monitor(MeasurementManagementObject,"Maximum_Separation");
group3.setDataGroupBatchSize(100);
group3.checkForReplicatedData();

// CREATE auto-timers for the groups

define auto3
auto3 = new IntervalSampler;
auto3.setGroup(group3);
auto3.setInterval_sec(30);  |
auto3.setSamplingOverlap(10);
auto3.setInitializationTime_sec(1800);

// 1 SO - 2 GO - 3 MMO FASTTIME/UNIFORM - 4 MMO ASYNCRONOUS

auto3.setSampleOption(3);
cemlist.add(auto3);

// add the external controller interface

define external
external = new SimController
external.setControlFileName("outputtest/controlSimC1.txt");
external.setObservationControlVariable(group3,"batchObservations");
cemlist.add(external)

sim.unpauseSimulation()
```

Figure 23: Sample Reconfigurable Flight Simulator Initialization Script

5.5 Simulation Diagnostic Testing

Several diagnostic tests are required before applying ranking and selection methods to these simulated configurations. First, the simulation, in this test case the RFS, must be validated as adequately mimicking the real-world system. In this case, validation was performed subjectively; extensive validations are often conducted for such simulations, but are beyond the scope of this study.

Next, a sampling rate must be found that provides observations exhibiting acceptable correlation. The performance, noted earlier in section 4.4.4, of ranking and selection methods, such as BGP4, highlight the level of acceptable correlation. Likewise, the batch size must be sufficiently large to ensure batched observations fit any normal distribution requirements of the ranking and selection methods. The remainder of this section highlights the application of these diagnostic tests to the RFS.

5.5.1 RFS Model Versus System Comparison

To demonstrate the ability of the RFS to mimic aircraft arrivals at an airport, a sample arrival configuration was developed for Atlanta International Airport. Aircraft in this configuration entered the Macey Two STAR with arrival densities based on historical data from 2002. Figure 24 highlights simulated arrivals by RFS. An overall aircraft inter-arrival time of 100 seconds mimicked 615 total arrivals observed during a standard operational day. Note arrivals are evenly distributed from the navigation aids Volunteer (VXV) and Spartanburg (SPA), and the "fix" AVERY for this experiment.

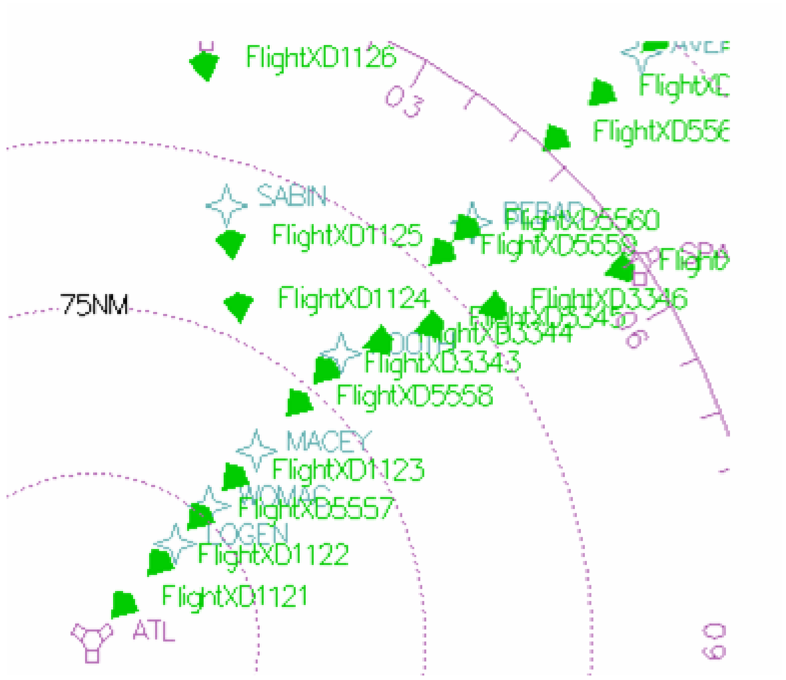


Figure 24: Simulation of Atlanta International Arrivals

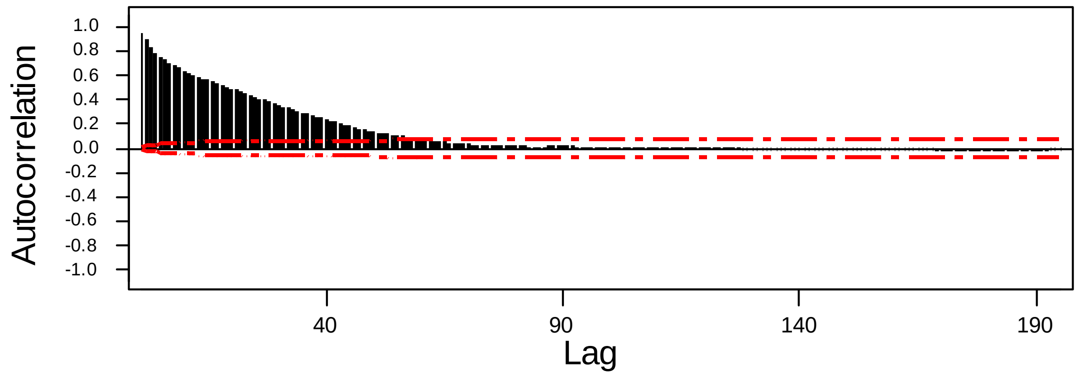
The goal of this diagnostic test was to subjectively validate simulated behavior as mimicking the real-world system. The use of a data analyzer agent verified obtained arrival rates were approximately the same as those specified. Also, a single long simulation run subjectively verified an absence of programming errors such as memory leaks. When using a simulation developed by several agencies with multiple contributors, a diagnostic test of this nature is necessary to ensure overall simulation stability. Note this diagnostic test did not validate controller strategies for aircraft spacing.

This diagnostic test also allowed for estimation of an appropriate simulation initialization period. Recall that sufficient simulation initialization is necessary to avoid bias in a steady-state simulation such as RFS. Here, the first aircraft arrived at ATL before approximately 30 simulation-minutes. Hence, data sampling starts after this initialization period for test case analysis.

5.5.2 RFS Simulation Output Correlation

In general, simulation output is correlated. Arrival data, such as average minimum separation in this test case, is highly correlated. Varying the observation sampling rate within RFS from 30 to 120 simulation seconds resulted in correlation coefficients ranging from 0.95 to 0.80 respectively. Note the increased computational requirement for obtaining decreased observation correlation. A side note, the sampling overlap was set to 10 seconds. The autocorrelation diagnostic test with a 30 second sampling rate is shown in Figure 25. Note observation correlation decreases as the time between observations increases.

Recall BGP4 achieved the desired probability in an Equal Spacing (ES) condition when the underlying autoregressive process was parameterized with $\phi = 0.95$. This corresponds approximately to data correlated with a coefficient value of 0.95. Hence, a 30 second sampling rate is appropriate if the competing configurations are assumed to be in the ES configuration.



Lag	Corr	T	LBQ	Lag	Corr	T	LBQ	Lag	Corr	T	LBQ	Lag	Corr	T	LBQ	Lag	Corr	T	LBQ
1	0.95	142.24	2.0E+04	16	0.55	20.33	1.8E+05	31	0.35	10.90	2.5E+05	46	0.18	5.26	2.7E+05	61	0.07	2.18	2.8E+05
2	0.90	80.29	3.8E+04	17	0.54	19.47	1.9E+05	32	0.34	10.46	2.5E+05	47	0.17	4.98	2.7E+05	62	0.07	2.02	2.8E+05
3	0.85	60.34	5.4E+04	18	0.52	18.65	1.9E+05	33	0.32	10.03	2.5E+05	48	0.16	4.74	2.7E+05	63	0.06	1.85	2.8E+05
4	0.79	49.27	6.9E+04	19	0.51	17.86	2.0E+05	34	0.31	9.60	2.6E+05	49	0.15	4.49	2.7E+05	64	0.06	1.69	2.8E+05
5	0.76	42.91	8.2E+04	20	0.50	17.14	2.1E+05	35	0.30	9.21	2.6E+05	50	0.14	4.26	2.7E+05	65	0.05	1.53	2.8E+05
6	0.73	38.32	9.4E+04	21	0.48	16.47	2.1E+05	36	0.29	8.83	2.6E+05	51	0.13	3.99	2.7E+05	66	0.05	1.39	2.8E+05
7	0.71	34.85	1.1E+05	22	0.47	15.82	2.2E+05	37	0.28	8.44	2.6E+05	52	0.13	3.76	2.7E+05	67	0.04	1.30	2.8E+05
8	0.69	32.03	1.2E+05	23	0.46	15.22	2.2E+05	38	0.26	8.04	2.6E+05	53	0.12	3.57	2.7E+05	68	0.04	1.21	2.8E+05
9	0.67	29.68	1.3E+05	24	0.44	14.62	2.3E+05	39	0.25	7.66	2.6E+05	54	0.11	3.39	2.8E+05	69	0.04	1.13	2.8E+05
10	0.65	27.71	1.4E+05	25	0.43	14.02	2.3E+05	40	0.24	7.31	2.7E+05	55	0.11	3.26	2.8E+05	70	0.04	1.07	2.8E+05
11	0.63	26.05	1.4E+05	26	0.42	13.46	2.3E+05	41	0.23	6.96	2.7E+05	56	0.10	3.08	2.8E+05	71	0.03	1.01	2.8E+05
12	0.61	24.59	1.5E+05	27	0.40	12.90	2.4E+05	42	0.22	6.61	2.7E+05	57	0.10	2.89	2.8E+05	72	0.03	0.95	2.8E+05
13	0.59	23.34	1.6E+05	28	0.39	12.34	2.4E+05	43	0.21	6.27	2.7E+05	58	0.09	2.70	2.8E+05	73	0.03	0.88	2.8E+05
14	0.58	22.23	1.7E+05	29	0.37	11.83	2.4E+05	44	0.20	5.93	2.7E+05	59	0.08	2.50	2.8E+05	74	0.03	0.80	2.8E+05
15	0.57	21.24	1.7E+05	30	0.36	11.35	2.5E+05	45	0.19	5.59	2.7E+05	60	0.08	2.35	2.8E+05	75	0.03	0.75	2.8E+05

Figure 25: RFS Autocorrelation Diagnostic Test

5.5.3 RFS Simulation Batched Observation Normality

Assuming a 30 second sampling rate is appropriate, the next simulation diagnostic involves determining the batch size for the Batch Means (BM) method. Recall a sufficiently large batch size results in normally distributed batched observations that are approximately uncorrelated. The key tradeoff in selecting a batch size is computational expense versus independent and normally distributed batch mean observations. A batch size of 100 resulted in observations described in Figure 26.

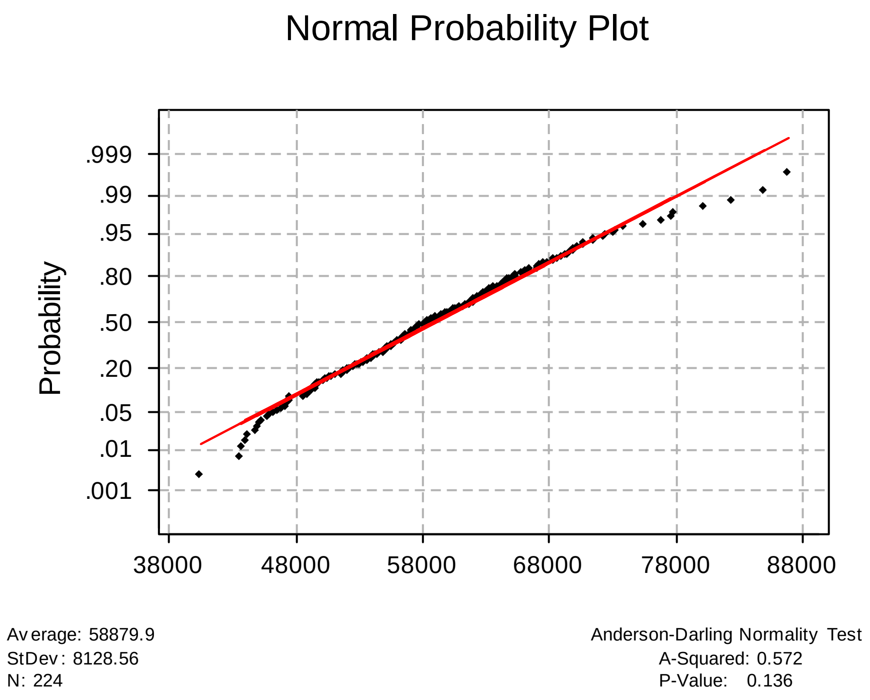


Figure 26: RFS Batched Observation Normality Diagnostic Test

A sampling rate of 30 seconds with a batch size of 100 obtains one batched mean observation for each 3000 simulation-seconds. With a dual 2.2 GHz processor workstation with 512 megabytes of RAM, this equates to approximately 180 seconds of computer-time. At p-values less than 0.136 there is no evidence the data is non-normal. For this reason, a batch size of 100 is selected for test case analysis.

5.5.4 RFS Simulation Batched Observation Variance Convergence

With an established sampling rate and batch size, the next diagnostic test focuses on batched observation variance convergence. Recall sufficient variance estimation is required for successful ranking and selection method performance. A single simulated configuration, shown in Figure 27, indicates batched observation variance does converge and how many batched observations are required for sufficient estimation.

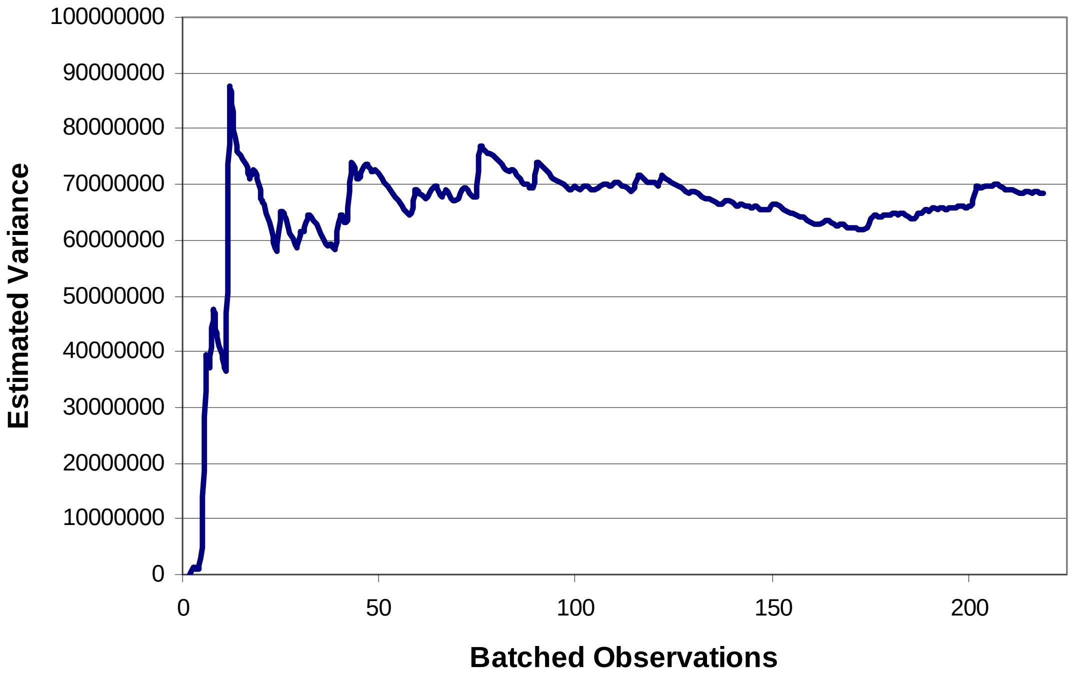


Figure 27: RFS Batch Mean Observation Variance Convergence Diagnostic

Note if this diagnostic test failed to indicate variance convergence then no known ranking and selection method could be used for comparative analysis because of the lack of variance consistency. Also, observe the convergence of variance is not smooth. Other long simulation runs should converge to the same value; however, the shape of the convergence curve could differ. In this example, it would be unwise to use BGP4 until at least 10 batched observations have been obtained due to a lack of variance convergence. With the given parameterization, this equates to 30000 simulation seconds or approximately 1800 computer seconds using a workstation with a dual 2.2 GHz processor. The practitioner must ensure some amount of variance of convergence while weighing the computational impact of delayed application of adaptive control techniques such as BGP4.

Convergence of Overlapping Batch Mean (OBM) observation variance is shown in Figure 28. BM and OBM variance estimators eventually converge to the same approximate value. Here, the practitioner should obtain at least 1000 overlapping batched observations before applying adaptive control techniques such as BGP4. This equates to 1099 unbatched observations from the simulation. While the unbatched observation requirement is significantly less for OBM than BM, the correlation of OBM observations is higher. The impact on adaptive control techniques of increased correlation from OBM observations is offset by the higher degrees of freedom of the variance estimator discussed in section 2.5.2. In general, the use of the OBM method should result in fewer unbatched observations than the BM method.

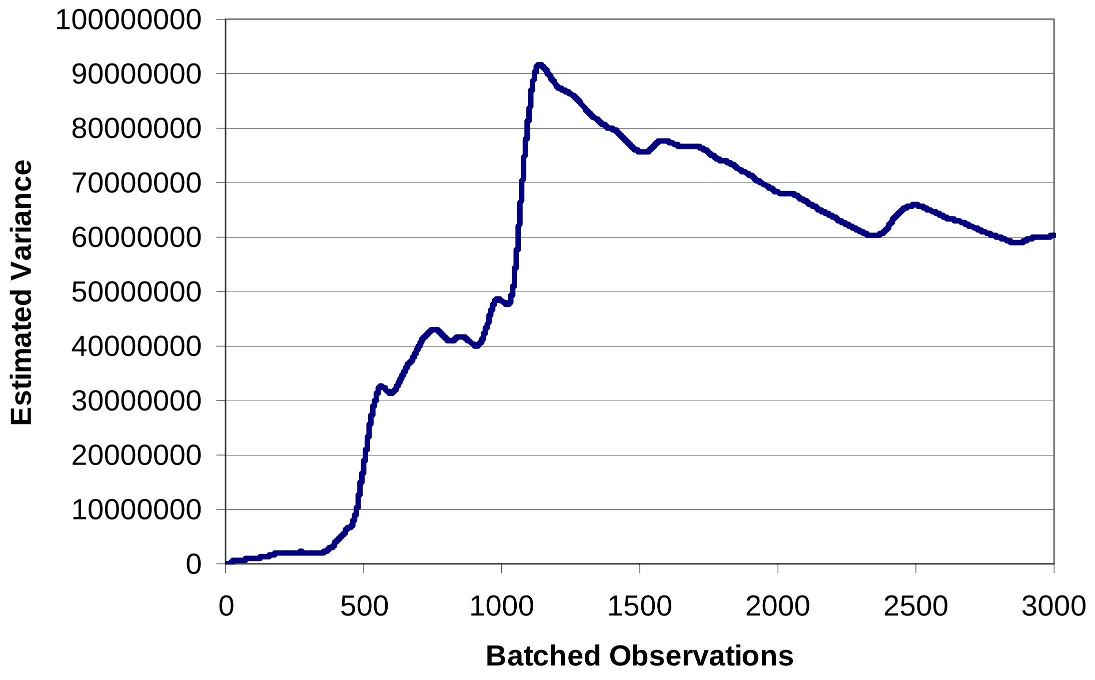


Figure 28: RFS Overlapping Batch Mean Observation Variance Convergence Diagnostic

5.5.5 Simulation Diagnostic Summary

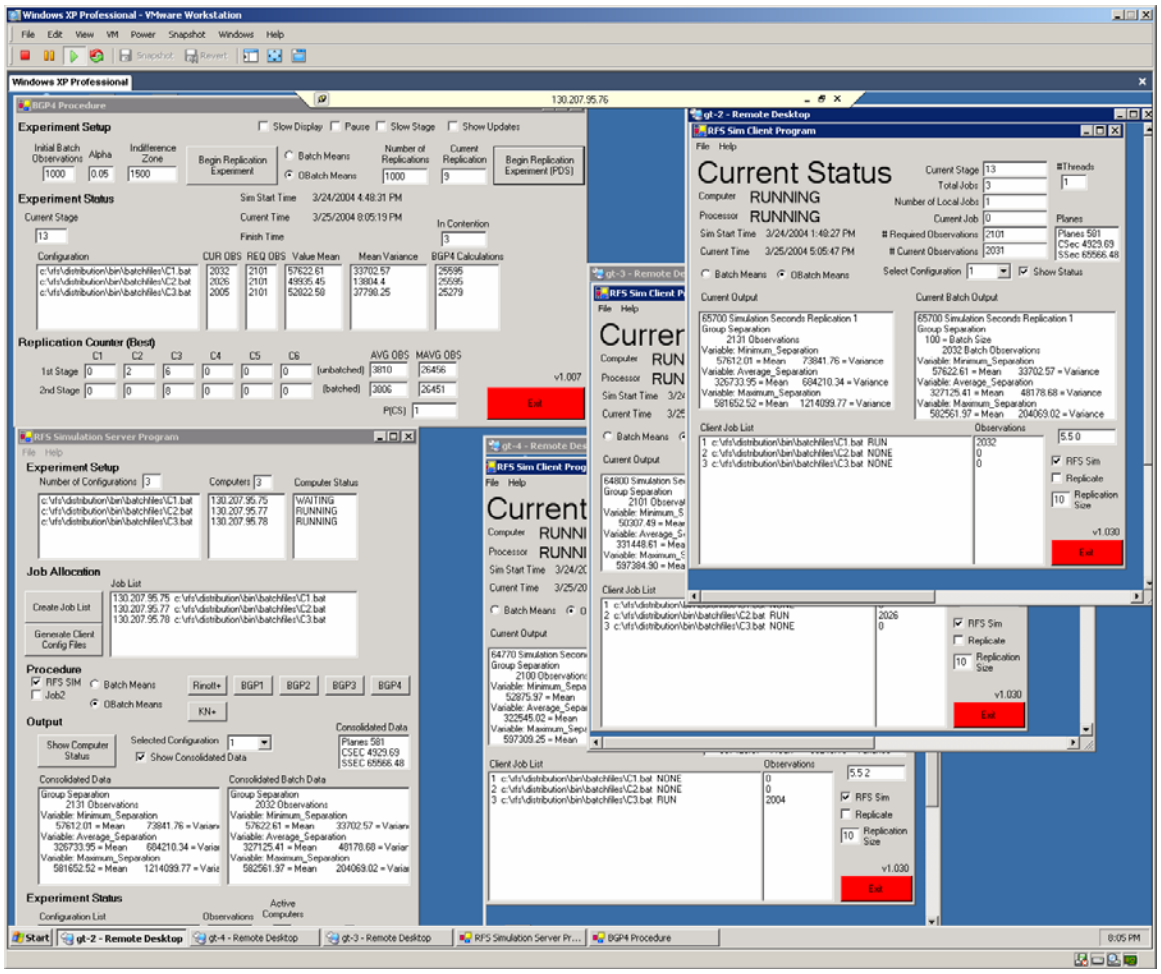
The diagnostics presented in this section can be generalized to any simulation where a unique metric defines performance. These diagnostics enable the application of ranking and selection methods, such as BGP4. First, the simulation must mimic the real-world system at a level subjectively accepted by the practitioner. This also allows for selection of a simulation initialization period. Next, a simulation sampling rate must be determined that results in data sufficiently uncorrelated for the ranking and selection method. Given a sampling rate, a batch size resulting in normally distributed observations is found. Lastly, variance convergence identifies the minimal number of initial observations necessary for the application of ranking and selection methods. Recall BGP4 significantly outperforms other known ranking selection methods when parameterized with the minimal number of initial observations. These diagnostics incur some initial computational expense; however, they ensure appropriate application of adaptive simulation control techniques.

5.6 Test Case Experiment

Given unlimited computational capacity, the designer would execute a large number of independent simulation replications to produce experimental results. However, the goal of adaptive control techniques, such as BGP4, is to reduce computational requirements for such assessments. A single experiment using BGP4 should provide rigorous statistical selection of the “best” simulated configuration. Multiple experiments can subsequently validate the comparative method. This section highlights a sample application of BGP4 to competing system configurations discussed in section 5.4.

5.6.1 Example Test Case Experiment

A snapshot is presented in Figure 29. In this example, three workstations contribute computational capacity while a fourth workstation functions as the server. The workstations for this experiment were homogeneous; workstation specifications include dual 2.2 GHz processors, 512 megabytes of RAM, and a Microsoft XP Professional operating system. Workstations communicate by operating system managed TCP/IP over a switched 100 megabit Ethernet connection. The right portion of the figure highlights participating workstations remotely mounted using Microsoft remote access software. Here, gt-2/3/4 workstations are the clients and gt-1 is the server or controller.


Windows XP Professional - VMware Workstation
File Edit View VM Power Snapshot Windows Help
Windows XP Professional
130.207.95.76

BGP4 Procedure

```
Experiment Setup          [ ] Slow Display  [ ] Pause  [ ] Slow Stage  [ ] Show Updates
Initial Batch Observations 1000   Alpha 0.05   Indifference Zone 1500
( ) Batch Means   (•) OBatch Means
Number of Replications 1000   Current Replication 9

Experiment Status
Current Stage 13          Sim Start Time  3/24/2004 4:48:31 PM
                          Current Time    3/25/2004 8:05:19 PM
                          Finish Time
In Contention 3

Configuration                              CUR OBS  REQ OBS  Value Mean  Mean Variance  BGP4 Calculations
c:\rfs\distribution\bin\batchfiles\C1.bat  2032     2101     57622.61    33702.57       25595
c:\rfs\distribution\bin\batchfiles\C2.bat  2026     2101     49935.45    13804.4        25595
c:\rfs\distribution\bin\batchfiles\C3.bat  2005     2101     52822.58    37798.25       25279

Replication Counter (Best)
            C1  C2  C3  C4  C5  C6               AVG OBS  MAVG OBS
1st Stage   0   2   6   0   0   0   (unbatched)  3810     26456
2nd Stage   0   0   8   0   0   0   (batched)    3806     26451
                                          P(CS)  1
v1.007
```

RFS Simulation Server Program

```
Experiment Setup
Number of Configurations 3        Computers 3       Computer Status
c:\rfs\distribution\bin\batchfiles\C1.bat   130.207.95.75   WAITING
c:\rfs\distribution\bin\batchfiles\C2.bat   130.207.95.77   RUNNING
c:\rfs\distribution\bin\batchfiles\C3.bat   130.207.95.78   RUNNING

Job Allocation
Job List
130.207.95.75  c:\rfs\distribution\bin\batchfiles\C1.bat
130.207.95.77  c:\rfs\distribution\bin\batchfiles\C2.bat
130.207.95.78  c:\rfs\distribution\bin\batchfiles\C3.bat

Procedure
[✓] RFS SIM    ( ) Batch Means
[ ] Job2       (•) OBatch Means

Output
Selected Configuration 1    [✓] Show Consolidated Data
Consolidated Data: Planes 581  CSEC 4929.69  SSEC 65566.48

Consolidated Data
Group Separation
      2131 Observations
Variable: Minimum_Separation
   57612.01 = Mean      73841.76 = Varian
Variable: Average_Separation
   326733.95 = Mean     684210.34 = Varia
Variable: Maximum_Separation
   581652.52 = Mean     1214099.77 = Vari

Consolidated Batch Data
Group Separation
      2032 Observations
Variable: Minimum_Separation
   57622.61 = Mean      33702.57 = Varian
Variable: Average_Separation
   327125.41 = Mean     48178.68 = Variar
Variable: Maximum_Separation
   582561.97 = Mean     204069.02 = Varia

Experiment Status
Configuration List    Observations    Active Computers
```

gt-2 - Remote Desktop — RFS Sim Client Program

```
Current Status
Computer   RUNNING
Processor  RUNNING
Sim Start Time  3/24/2004 1:49:27 PM
Current Time    3/25/2004 5:05:47 PM
Current Stage 13   Total Jobs 3   Number of Local Jobs 1   Current Job 0
# Required Observations 2101   # Current Observations 2031
#Threads 1
Planes: Planes 581  CSec 4929.69  SSec 65566.48
( ) Batch Means  (•) OBatch Means   Select Configuration 1   [✓] Show Status

Current Output
65700 Simulation Seconds Replication 1
Group Separation
      2131 Observations
Variable: Minimum_Separation
   57612.01 = Mean      73841.76 = Variance
Variable: Average_Separation
   326733.95 = Mean     684210.34 = Variance
Variable: Maximum_Separation
   581652.52 = Mean     1214099.77 = Variance

Current Batch Output
65700 Simulation Seconds Replication 1
Group Separation
   100 = Batch Size
      2032 Batch Observations
Variable: Minimum_Separation
   57622.61 = Mean      33702.57 = Variance
Variable: Average_Separation
   327125.41 = Mean     48178.68 = Variance
Variable: Maximum_Separation
   582561.97 = Mean     204069.02 = Variance

Client Job List                                  Observations
1  c:\rfs\distribution\bin\batchfiles\C1.bat  RUN    2032
2  c:\rfs\distribution\bin\batchfiles\C2.bat  NONE   0
3  c:\rfs\distribution\bin\batchfiles\C3.bat  NONE   0
5.5.0   [✓] RFS Sim   [ ] Replicate   10 Replication Size
v1.030
```

gt-3 - Remote Desktop — RFS Sim Client Program

```
Client Job List                                  Observations
2  c:\rfs\distribution\bin\batchfiles\C2.bat  RUN    2026
3  c:\rfs\distribution\bin\batchfiles\C3.bat  NONE   0
[✓] RFS Sim   [ ] Replicate   10 Replication Size
v1.030
```

gt-4 - Remote Desktop — RFS Sim Client Program

```
Client Job List                                  Observations
1  c:\rfs\distribution\bin\batchfiles\C1.bat  NONE   0
2  c:\rfs\distribution\bin\batchfiles\C2.bat  NONE   0
3  c:\rfs\distribution\bin\batchfiles\C3.bat  RUN    2004
5.5.2   [✓] RFS Sim   [ ] Replicate   10 Replication Size
v1.030
```

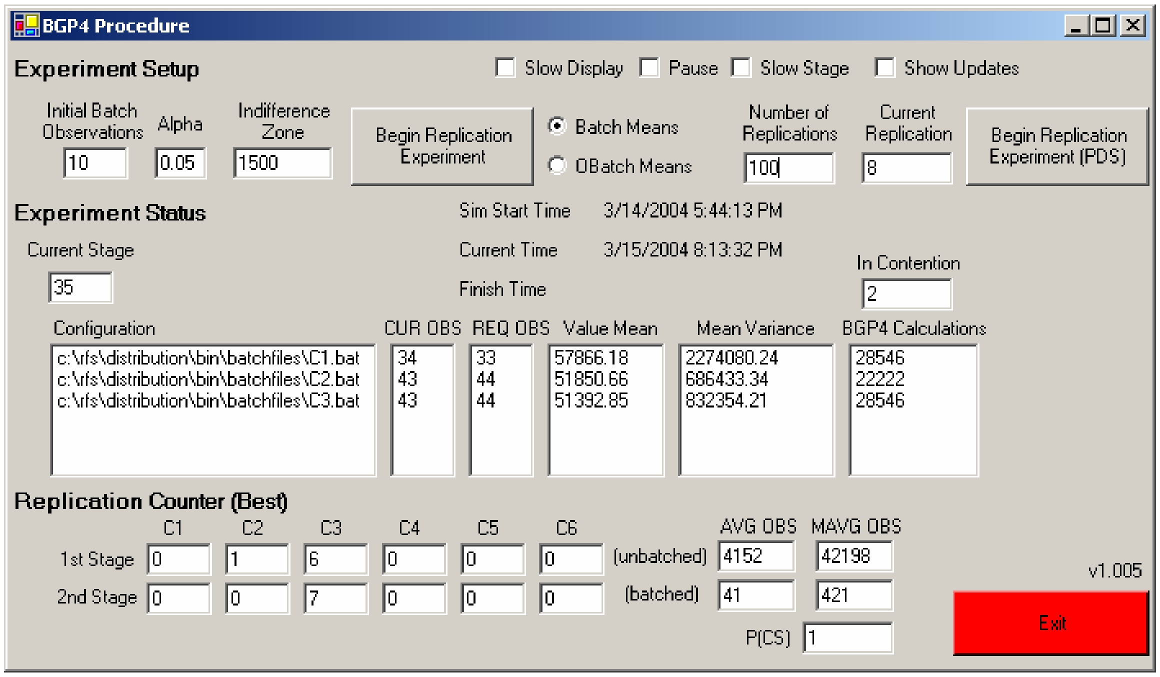
Figure 29: Snapshot of Test Case Experiment Session

Each contributing workstation executes a client module that controls the simulated configurations and communicates both status and estimated data parameters to the controller. The server samples and consolidates data in the manner prescribed by the designer. In this example, overlapping batch means obtain observations for the control technique; the DA samples every 30 simulation seconds with an overlap of 10 seconds. BGP4 is the adaptive control technique for this example. At this point in the experiment, all configurations are still in contention for selection as the “best”.

The goal of this sample experiment was to highlight stable operation between the client and server modules demonstrated by no loss of control by the server of a client simulation and also by data streams updated without error. This experiment also provided verification, by separate analysis, of embedded statistical collection methods. Lastly, this sample experiment highlights the functionality of combined embedded statistical estimators, PDS techniques, and ranking and selection methods.

An example of adaptive control technique performance is shown in Figure 30 for BGP4. In this example, batch means were used to obtain observations. Observe configuration C1 was eliminated from further consideration at 33 batched observations but 34 observations were actually obtained from the participating workstation. This exemplifies the RS screening process eliminating a competing configuration from further analysis. PDS error handling allows for further observation acquisition should a communication error occurs, hence the 34th observation.

Experiment metrics, such as the estimated probability of correct selection and required observations, are maintained by the controller as shown in the bottom right of Figure 30. Performance of BM and OBM methods can be compared by the number of raw observations. Additionally, the use of varying random number seeds facilitates comprehensive analysis of the simulated configurations. The computational tractability of this approach is addressed in following sections.


BGP4 Procedure

Experiment Setup

☐ Slow Display ☐ Pause ☐ Slow Stage ☐ Show Updates

Initial Batch Observations	Alpha	Indifference Zone			Number of Replications	Current Replication	
10	0.05	1500	Begin Replication Experiment	[● Batch Means] ○ OBatch Means	100	8	Begin Replication Experiment (PDS)

Experiment Status

Current Stage: 35

Sim Start Time 3/14/2004 5:44:13 PM

Current Time 3/15/2004 8:13:32 PM

Finish Time

In Contention: 2

Configuration	CUR OBS	REQ OBS	Value Mean	Mean Variance	BGP4 Calculations
c:\rfs\distribution\bin\batchfiles\C1.bat	34	33	57866.18	2274080.24	28546
c:\rfs\distribution\bin\batchfiles\C2.bat	43	44	51850.66	686433.34	22222
c:\rfs\distribution\bin\batchfiles\C3.bat	43	44	51392.85	832354.21	28546

Replication Counter (Best)

	C1	C2	C3	C4	C5	C6		AVG OBS	MAVG OBS
1st Stage	0	1	6	0	0	0	(unbatched)	4152	42198
2nd Stage	0	0	7	0	0	0	(batched)	41	421

P(CS) 1

v1.005

Exit

Figure 30: BGP4 Sample Test Case Application

5.6.2 Best Case Arrival Routing Analysis

The "best" or "preferred" arrival routing configuration obtains the highest average minimum separation for a given total throughput. From RFS test diagnostics a 30 second sampling rate, a 10 second sampling overlap, and batch size of 100 is used to acquire observations using the batch means method. This experiment also provides validation of the combined embedded statistical encapsulation methods, adaptive control techniques, and distributed simulation architecture. Here, validation is obtained by separate analysis of long simulated runs of the competing system configurations.

Table 19 presents experimental results from the application of BGP4 to configurations described in section 5.4. Experiment duration for these specific random number seeds was approximately 96 computer-minutes using four dual 2.2 GHz processor workstations each with 512 megabytes of RAM. The workstations communicated by operating system managed TCP/IP over a 100 megabit Ethernet connection. Three of the workstations provided computational capacity while the fourth acted as the controller in this experiment. For this experiment, simulation seeds for aircraft generation remained constant between experimental runs to verify the underlying simulation, i.e. the RFS, was capable of reproducible results. Note BGP4 is intended for needing only one experiment to select the "best" competing system configuration.

Table 19: Arrival Routing Comparison of Average Minimum Separation in Feet

	Average Minimum Separation (feet)							
	Replication							
Configuration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C1	**56652**	**56652**	**56928**	**56856**	**56652**	**56856**	**56652**	**56652**
C2	51031	51031	51031	51066	51031	51066	51031	51031
C3	50693	50693	50504	50504	50693	50589	50693	50693

Configuration C1 was selected as the "best" for all experimental replications. Observe the average minimum separation varies between experiments. This is attributed to communication errors between contributing workstations and the controller. Recall the distributed simulation architecture allows contributing workstations to continue observation acquisition when a communication error is encountered. Each communication error extended experiment execution time by the time required to obtain an additional batched observation, in this case approximately three minutes of computer-time.

5.6.3 Worst Case Arrival Routing Analysis

The "worst" arrival routing configuration obtains the lowest average minimum separation for a given total throughput. Experimental parameterization is the same as the previous experiment. Table 20 highlights the application of BGP4 to identify the "worst" arrival route density allocation from a single experiment. Configuration C3 is the "worst". Recall BGP4 ensures statistical rigor in the selection of a competing simulated configuration from a single experiment.

Table 20: Worst Case Arrival Routing Comparison

	Configuration		
	C1	**C2**	**C3**
Average Minimum Separation (feet)	59186	53562	**50165**

Replicating this experiment validates the comparative method and also allows for performance estimation. Table 21 presents the application of BGP4 to identify the "worst" arrival route density allocation. Initial random number seeds varied in an incremental manner controlled by the server module for these experiments. Twenty-one experimental replications provided estimators of average minimum separation, standard error, and the average required number of unbatched observations.

Table 21: Replicated Worst Case Arrival Routing Comparison

Average Minimum Separation			
	Configuration		
	C1	**C2**	**C3**
Mean (feet)	57569	52174	**49706**
SE	232	148	155
$\hat{\hat{T}}$	2700	5600	5600

Configuration C3, i.e., the highest route densities on the two southern paths in the Macey Two STAR, is confirmed to be the "worst" performing simulated configuration.

The probability for selecting configuration C3 equaled 1.00. Also, the standard error indicates the competing configurations are statistically differentiable in post hoc analysis. Observe the early elimination of configuration C1 from comparative analysis shown by the low average number of unbatched observations, $\hat{\bar{T}}$. Configurations C2 and C3, the last two competing configurations under analysis, terminate at the same number of unbatched observations when one of them is selected as the “worst”.

5.6.4 Routing Analysis Summary

Application of combined adaptive control and PDS techniques is appropriate after RFS diagnostics. In this test case, the applied technique selected configuration C1 as the “best” and C3 as the “worst” route allocation scenario. Specifically, configurations with high arrival route densities on southern paths into Atlanta International Airport resulted in the lowest average minimum separation. Several replications of the experiment with different random number seeds demonstrate that considered configurations were statistically differentiable. This leads to a definitive conclusion that configuration C3 is indeed the worst possible routing scenario.

Although not observed in the previous experiments, if the competing configurations were not statistically differentiable then replicated experiments could result in several configurations being identified as the “best” or “worst”. Technically, the practitioner would fail to achieve the desired probability. However, this highlights the robustness of ranking and selection methods. Recall the indifference-zone parameter is a subjective level where the practitioner would not discriminate between competing configurations. For this test, the parameter is 1500 feet which equates to approximately six seconds of flight time. If several configurations are selected as the “best” or “worst” after replicated experiments, then the practitioner should interpret these results in a grouped nature. For example, if any two configurations were consistently identified as the “worst” configurations then the practitioner may collectively group these configurations as poor performers.

5.7 Test Case Computational Performance Comparison

The goal of integrated adaptive control and PDS techniques is statistical selection that is efficient and correct. This section explores the computational performance efficiency of two job allocation schemes, i.e., assignment of simulated configuration observation requirements to contributing workstations. Also, the effect of Batch Mean (BM) and Overlapping Batch Mean (OBM) batched observation acquisition methods on the total number of unbatched observations is explored.

Partial Job Allocation (PJA) entails sequential job allocation to contributing workstations. For example, if there are three participating workstations and three configurations then a single job is allocated to each workstation. As another example, if there are three workstations and four configurations then a single job is allocated to two of the participating workstations. The third workstation is assigned two jobs. PJA is relevant when the number of configurations in contention is greater than or equal to the number of participating workstations.

Full Job Allocation (FJA) consists of all configurations being distributed to all participating workstations. Implementation of FJA assumes the combination of data from similarly configured simulations is appropriate as discussed in section 2.2.3. The primary motivation for FJA stems from the screening phase of discussed adaptive control techniques. Specifically, the elimination of a competing configuration may result in workstation idleness during an experiment.

Table 22 highlights the application of BGP4 to the configurations discussed in section 5.4. Average minimum separation is reported along with the standard error and the estimated number of unbatched observations. Configuration C3 is identified as the "worst" in all conditions. The estimated probability of correction selection, $\hat{P}(CS)$, equaled 1.00 for all conditions. Also, mean estimators for each configuration are not statistically differentiable by the job allocation or batching method. Observe the standard error is higher for OBM and even higher for full job allocation. There appears to be higher estimated variance for these methods. Note the decreased unbatched observation requirement, $\hat{\hat{T}}$, for OBM versus BM. There is also a slightly increased unbatched observation requirement using full job allocation versus partial job allocation.

Table 22: Test Case Computational Analysis

	Partial Job – BM			**Full Job – BM**			**Partial Job – OBM**		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**
Mean (feet)	57275	52061	**49614**	57051	52467	**49802**	57472	52263	**49028**
SE	344	208	216	482	338	306	384	214	370
$\hat{\hat{T}}$	2700	5600	5600	2800	6000	6000	2173	4918	4933

While the high $\hat{P}(CS)$ is encouraging, it fails to validate inherent assumptions in full job allocation. Instead, the high $\hat{P}(CS)$ for these replicated experiments implies a relatively large indifference-zone parameter. Recall the indifference-zone parameter is subjectively selected by the practitioner.

Figure 31 graphically highlights the estimated NOW minutes required for an experiment. Clearly, the observation or computational requirement is directly related to execution time. Observe FJA completes experiments faster than PJA because this job allocation scheme avoids workstation idleness. A NOW minute in this context is the collective contributions from four dual 2.2 GHz processor workstations for one minute apiece. Each workstation possessed 512 megabytes of RAM and communicated through a 100 megabit Ethernet switch on operating system managed TCP/IP. Three of the workstations contribute computational capacity while the fourth acts as the controller.

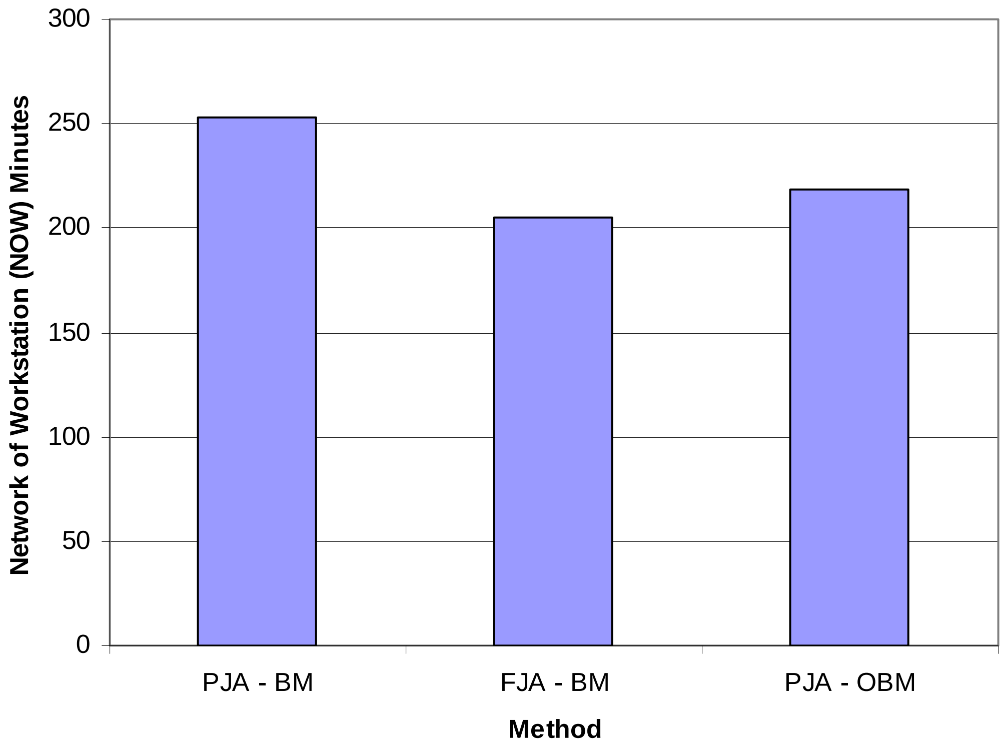


Figure 31: Test Case Computational Performance

5.8 Test Case Summary

The application of adaptive control and distributed simulation techniques to an existing large-scale simulation is relatively new. The required effort to modify an existing simulation, in this case the Reconfigurable Flight Simulator, for the application of these techniques can be relatively small compared to the overall development of the simulation itself. The application of these techniques also avoids post hoc analysis.

Simulation diagnostics ensure the appropriate application of these techniques. Results from this test case, for example, imply general procedures for Atlanta International Airport. Amongst the configurations tested here, equal routing densities for the Macey Two STAR provides the largest average minimum separation of aircraft. Likewise, high route densities for southern paths on the STAR should be avoided.

Computational savings from the integration of adaptive control and distributed simulation techniques are potentially large. A distributed simulation architecture in the form of a network of workstations provides approximately linear increases in overall experimental performance for a small number of workstations. Without adaptive control techniques, long runs of each configuration are needed for appropriate statistical analysis. In a practical sense, data analysis as a post hoc activity also requires personnel-hours and the input/output burden or recording. This scenario was not used for baseline comparison as it can be assumed to be significantly long.

CHAPTER 6

CONCLUSIONS

This research brings together the fields of simulation, embedded statistical analysis, adaptive control techniques, parallel and distributed simulation, and complex system analysis. Combination of these fields itself is an intellectual contribution; in addition, there are several derived practical and theoretical contributions within each field. Practical modifications to existing large-scale simulations for these techniques establish a methodology suitable for reuse. Embedded statistical calculations enable runtime analysis. Theoretical development of adaptive control techniques, such as ranking and selection, combined with the practical aspects of the implementation, create an analytic environment that allows discrimination between competing system configurations. PDS methods offer near linear reduction in terms of computational expense for a small number of participating workstations.

In a unified sense, this research enables enhanced use of simulation in the design and analysis of complex systems. Ranking and selection methods provide a control technique. Embedded statistical analysis allows for runtime input to this control technique. PDS implementation provides the computational capacity necessary for practical use. Modifications and extensions of these somewhat related disciplines establish a coherent analysis tool for sophisticated design activities.

Embedded statistical estimators incur computational overhead. However, this overhead is justified by the utility of these estimators in enabling runtime comparison between competing simulated system configurations. Incorporated with ranking and selection methods, these estimators enable calculation of the number of required observations necessary for rigorous statistical analysis.

Appropriate combination of simulation output from these embedded statistical estimators must address several issues. First, generated random numbers must be sufficiently offset to avoid redundant observations. Second, a small number of combined observations may result in underestimation of the variability of the simulated process. Also, combination of simulation output must represent behavior possible from the actual system, such as combining seasonal or related elements of a simulated configuration.

Embedded statistical analysis enables the application of Ranking and Selection (RS) methods for comparison of competing simulated system configurations. Embedded statistical estimators bound the variance central to RS method calculations. Specifically, selection of the initial number of observations directly affects current method performance in terms of the total required observations. This motivated the development of the RS BGP4 method, a new RS method enabled by the incorporation of embedded statistical estimators. While derived from the KN+ RS method, BGP4 conforms to the underlying simulated process through the use of embedded statistical estimators.

Diagnostics to ascertain the appropriate application of BGP4 can be generalized to any large-scale simulation. First, determining where data sampling may begin avoids initialization bias. Next, underlying process serial correlation can be mitigated by the determination of a sufficiently long sampling rate. Then, a batch size can be found that results in sufficiently normal batched observations. Lastly, the convergence of variance estimators gives a lower bound on when the RS method may be employed.

BGP4 performs significantly better than other RS methods in terms of computational requirements while still achieving the desired probability. BGP4 was found to perform well with mildly or moderately correlated data when competing simulations are in the Least Favorable configuration of means. BGP4 also performed well on highly correlated data when competing simulations are in the Equal Spacing configuration of means.

Incorporation of BGP4 within a distributed simulation architecture using a Network of Workstations (NOW) increases computational capacity. A key issue for the application of a distributed simulation architecture is job allocation. Sequential allocation results in workstation idleness. Allocation of all simulation jobs to all participating workstations is only appropriate when simulation output may be combined for analysis.

Application of BGP4 within this distributed simulation architecture to an existing large-scale simulation produced promising results. Modifications to an existing large-scale hybrid simulation, in this case the Reconfigurable Flight Simulator, were small compared to development of the simulation itself. Initial application of these techniques with partial job allocation while using Batch Means for observation acquisition identified the "best" and "worst" competing configurations. Computational performance was improved by full job allocation and the Overlapping Batch Means (OBM) data acquisition methods.

Experimental results highlight the types of insights that this method provides. In this test case, allocation of arrivals on the two southern paths of the Macey Two STAR was found to significantly lower average minimum separation distance, with implications for arrival procedures into Atlanta International Airport.

This comparative method may be applied to modeled complex systems differentiated by a unique performance metric. For example, continuous-valued profit measures or the relative physical separation of modeled components could identify desired performance for a particular system. The comparative method can also incorporate derived metrics from discrete variables, such as average throughput. The described method relies on observations of estimated mean values. These observations must exhibit characteristics enabling the application of ranking and selection methods. Specifically, ranking and selection method performance must be robust for the achieved normality and serial correlation of these observations. Lastly, obtaining observations must be computationally tractable.

6.1 Contribution Summary

Contributions of this research can be summarized as follows:

- Integration of embedded statistical analysis, ranking and selection methods, and parallel and distributed simulation techniques for the analysis of complex systems.
- Embedded data estimators that are both efficient and accurate.
- Extended ranking and selection methods that determine which simulated configurations are still in contention for selection as the “best” along with the number of required observations required. Specifically, BGP4 only requires preliminary diagnostics to ascertain the initialization period, sampling rate providing acceptable correlation, batch size providing sufficiently normal observations, and variance convergence.

- A generalized method for diagnosing simulation parameters needed for appropriate application of adaptive control and distributed simulation techniques.
- Specific results on arrival procedures for Atlanta International Airport as a demonstration of the insight these techniques can provide.

6.2 Future Efforts

Several extensions may be possible of the methods and techniques developed in this effort. First, embedding statistical analysis within an existing simulation involves a relatively small expenditure in terms of personnel-hours compared to the reduction in required post hoc analysis. The embedded statistical methods developed here should be explored to ascertain other potential applications in the use of simulation as a design activity.

BGP4 is a significant extension to current RS methods. In certain situations, it may be assumed that simulation observations are related between configurations. In such cases, a paired t-test may provide a stricter statistical comparison. Incorporation of this test statistic for comparative analysis is bounded by the multivariate normal test statistics used by BGP4. Potentially, the incorporation of a t-test, under assumed conditions, would increase the computational efficiency of BGP4.

Another RS method extension would incorporate nonparametric test statistics. The techniques developed here assume normally distributed observations. Nonparametric test statistics preclude a reliance on the assumption of normality in a general sense. Incorporation of nonparametric test statistics would be relatively easy with the given techniques. It is possible the incorporation of nonparametric test statistics would speed the acquisition of observations with an overall increase of computational efficiency.

Job queuing for participating workstations and the overall experimental architecture can be improved in several areas. If participating workstations have heterogeneous performance, then allowing execution of simulated configurations scaled to the speed of the workstation can be more efficient. If the length of a job is known, job shop like algorithms may be employed to reduce computational expense. The central issues here are simulation initialization and state space transfer between contributing workstations.

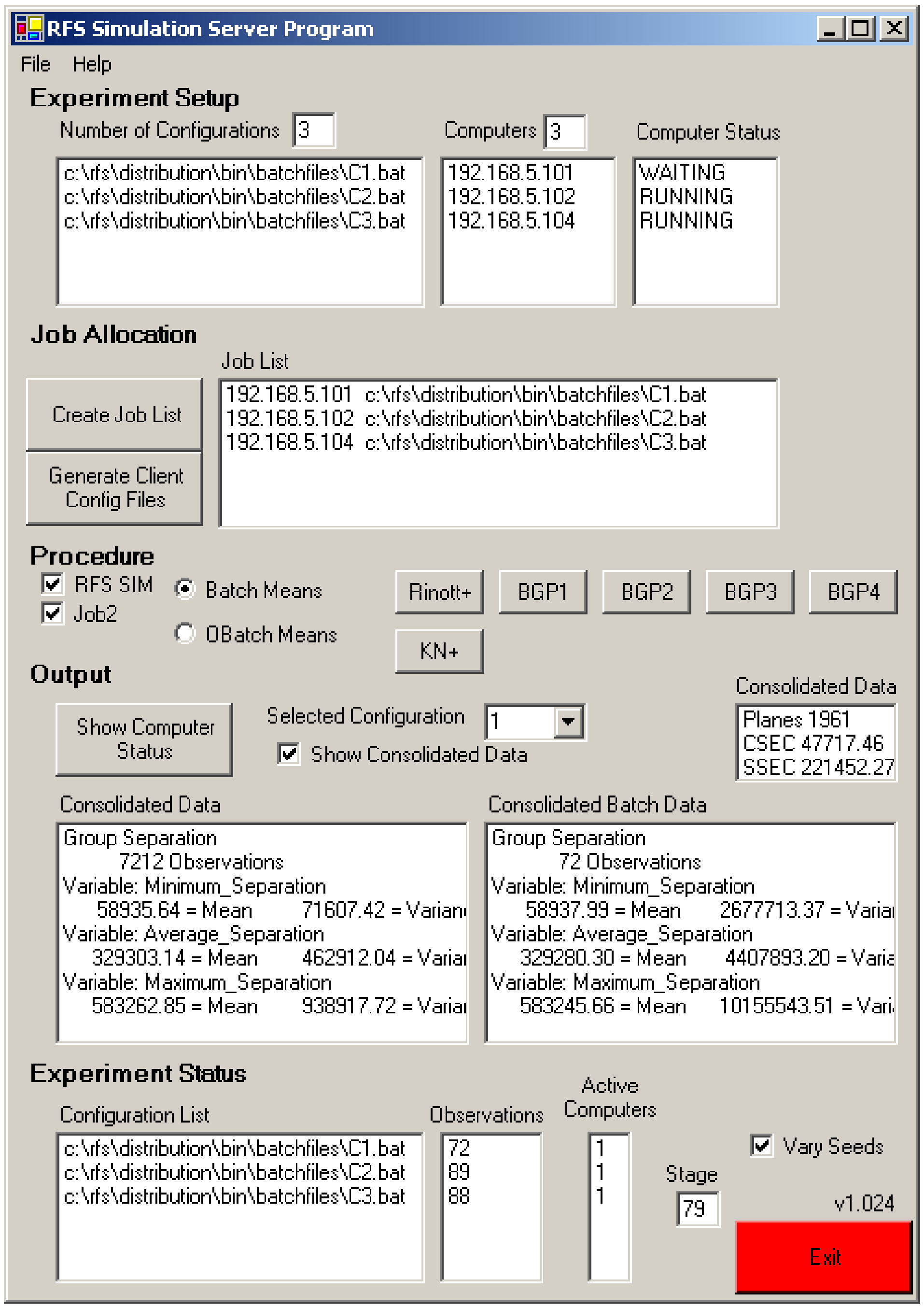
Integration of these techniques extends the use of simulation as an analysis and design activity for complex systems. Application of developed techniques to other large-scale simulations should be explored. One interesting application is rare event analysis. Rather than focusing on estimated mean values, rare events are often minima or maxima. Incorporation of test statistics suited for extreme value analysis would extend the applicability of this comparative method.

APPENDIX A: DISTRIBUTED SIMULATION SOFTWARE

The distributed simulation architecture developed for this research uses a Network of Workstations (NOW) to acquire computational capacity. Client-server software enables control of participating workstations in an experiment. The server module controls all aspects of the experiment. The client module acts as an interface between the server and simulated configurations. Ranking and Selection (RS) method modules apply adaptive techniques to experiment execution.

The server is initialized by a runtime interpreted script. This script identifies participating workstations by network address. The location of the existing simulation executable program is also contained in this script. Note the simulation executable programs used in this research are also initialized by runtime interpreted scripts detailing specific configuration information. Here, an executable batch file associates the existing simulation executable program with a specific configuration. Additionally, the server initialization script identifies the location of simulated configuration output.

A snapshot of the server module is shown in Figure 32. In this example, three workstations participate in an experiment entailing the comparison of three simulated configurations. At the time of the snapshot, one workstation is waiting, or idle, for further commands. A workstation becomes idle after completing all jobs. Recall a job denotes the acquisition of a specified number of observations from a simulated configuration. Here, all clients are assigned all configurations. Simulation output is viewable in the bottom of the module. Note the batch means method for this experiment. Overall experiment status is highlighted by the number of obtained observations.



Experiment Setup

Number of Configurations: 3

Computers: 3

```
c:\rfs\distribution\bin\batchfiles\C1.bat    192.168.5.101    WAITING
c:\rfs\distribution\bin\batchfiles\C2.bat    192.168.5.102    RUNNING
c:\rfs\distribution\bin\batchfiles\C3.bat    192.168.5.104    RUNNING
```

Job Allocation

Job List

```
192.168.5.101  c:\rfs\distribution\bin\batchfiles\C1.bat
192.168.5.102  c:\rfs\distribution\bin\batchfiles\C2.bat
192.168.5.104  c:\rfs\distribution\bin\batchfiles\C3.bat
```

Procedure

[✓] RFS SIM
[✓] Job2
[✓ Batch Means]
OBatch Means

Output

Selected Configuration: 1
[✓] Show Consolidated Data

Consolidated Data

```
Planes 1961
CSEC 47717.46
SSEC 221452.27
```

Consolidated Data

```
Group Separation
        7212 Observations
Variable: Minimum_Separation
    58935.64 = Mean        71607.42 = Varian
Variable: Average_Separation
    329303.14 = Mean       462912.04 = Varia
Variable: Maximum_Separation
    583262.85 = Mean       938917.72 = Varia
```

Consolidated Batch Data

```
Group Separation
        72 Observations
Variable: Minimum_Separation
    58937.99 = Mean       2677713.37 = Varia
Variable: Average_Separation
    329280.30 = Mean      4407893.20 = Varia
Variable: Maximum_Separation
    583245.66 = Mean      10155543.51 = Vari
```

Experiment Status

Configuration List	Observations	Active Computers
c:\rfs\distribution\bin\batchfiles\C1.bat	72	1
c:\rfs\distribution\bin\batchfiles\C2.bat	89	1
c:\rfs\distribution\bin\batchfiles\C3.bat	88	1

Stage: 79

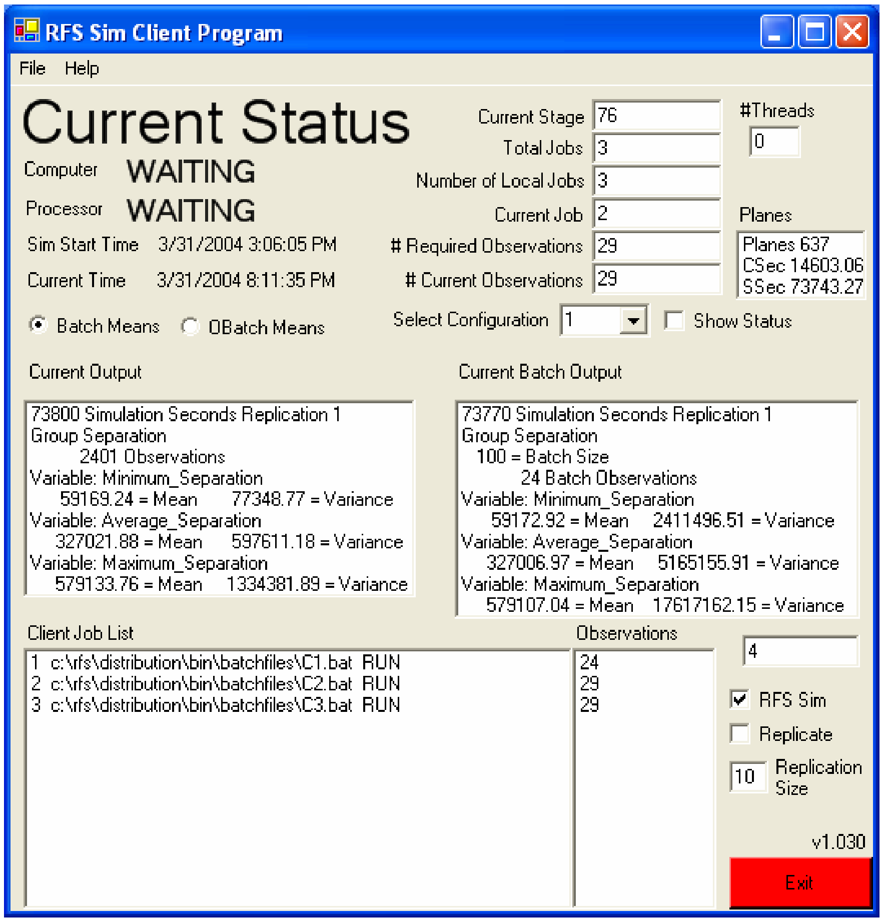
[✓] Vary Seeds

v1.024

Figure 32: Distributed Simulation Server Module

The server issues commands to participating workstations via script files. Each workstation runs a client module providing an interface between the server and simulated configurations. This script files contains the location of the batch files, output, and the number of required observations for the client. Note the client schedules jobs sequentially. The client issues commands to the existing simulation executable program by script files. Recall existing simulations can interpret commands of "PAUSE", "UNPAUSE", and "TERMINATE". Also, existing simulations are capable of embedded statistical analysis.

A snapshot of the client module is shown in Figure 33. In this example, the client is waiting for further commands from the server. Here, the client has completed three jobs. Sample simulated configuration output allows for experimental monitoring. Lastly, the number of observations highlights this workstations contribution to the experiment.



Current Status

Computer WAITING

Processor WAITING

Sim Start Time 3/31/2004 3:06:05 PM

Current Time 3/31/2004 8:11:35 PM

[✓ Batch Means] OBatch Means

Current Stage 76

Total Jobs 3

Number of Local Jobs 3

Current Job 2

Required Observations 29

Current Observations 29

Select Configuration 1

Show Status

#Threads 0

Planes

Planes 637
CSec 14603.06
SSec 73743.27

Current Output

```
73800 Simulation Seconds Replication 1
Group Separation
       2401 Observations
Variable: Minimum_Separation
    59169.24 = Mean      77348.77 = Variance
Variable: Average_Separation
    327021.88 = Mean    597611.18 = Variance
Variable: Maximum_Separation
    579133.76 = Mean   1334381.89 = Variance
```

Current Batch Output

```
73770 Simulation Seconds Replication 1
Group Separation
  100 = Batch Size
       24 Batch Observations
Variable: Minimum_Separation
    59172.92 = Mean    2411496.51 = Variance
Variable: Average_Separation
    327006.97 = Mean    5165155.91 = Variance
Variable: Maximum_Separation
    579107.04 = Mean   17617162.15 = Variance
```

Client Job List

```
1  c:\rfs\distribution\bin\batchfiles\C1.bat  RUN
2  c:\rfs\distribution\bin\batchfiles\C2.bat  RUN
3  c:\rfs\distribution\bin\batchfiles\C3.bat  RUN
```

Observations

```
24
29
29
```

4

[✓ RFS Sim]

Replicate

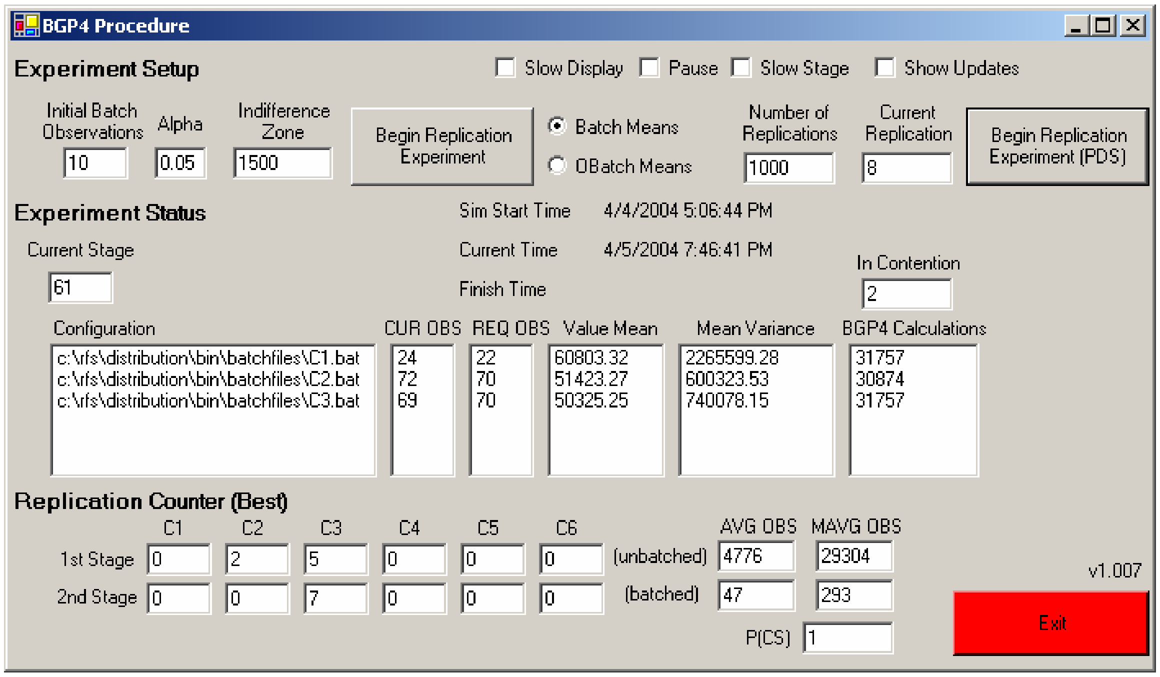
10 Replication Size

v1.030

Figure 33: Distributed Simulation Client Module

RS method modules apply adaptive techniques to an experiment by determining the number of required observations and which simulated configurations are still in contention for selection as the “best”. The server module uses this information to determine job requirements at each stage of the experiment. Here, a stage denotes the execution of all jobs. Recall RS methods terminate when either the upper bound on the number of observations is achieved or there is only one configuration still in contention.

A snapshot of the BGP4 RS method module is shown in Figure 34. The practitioner sets RS method parameters to include the initial number of observations, desired probability, and indifference-zone parameter. Also, the number of experiment replications is set. In this example, the eighth experiment is currently proceeding with two configurations still in contention for selection as the “best”. Configuration specific data facilitates experiment monitoring. Note configuration C3 was selected as the “best” in the seven completed experiments. Observation requirements for an experiment give insight on the computational expense.


BGP4 Procedure

Experiment Setup

☐ Slow Display ☐ Pause ☐ Slow Stage ☐ Show Updates

Initial Batch Observations	Alpha	Indifference Zone	Number of Replications	Current Replication
10	0.05	1500	1000	8

Begin Replication Experiment

[✓ Batch Means] OBatch Means

Begin Replication Experiment (PDS)

Experiment Status

Current Stage: 61

Sim Start Time 4/4/2004 5:06:44 PM

Current Time 4/5/2004 7:46:41 PM

Finish Time

In Contention: 2

Configuration	CUR OBS	REQ OBS	Value Mean	Mean Variance	BGP4 Calculations
c:\rfs\distribution\bin\batchfiles\C1.bat	24	22	60803.32	2265599.28	31757
c:\rfs\distribution\bin\batchfiles\C2.bat	72	70	51423.27	600323.53	30874
c:\rfs\distribution\bin\batchfiles\C3.bat	69	70	50325.25	740078.15	31757

Replication Counter (Best)

	C1	C2	C3	C4	C5	C6		AVG OBS	MAVG OBS
1st Stage	0	2	5	0	0	0	(unbatched)	4776	29304
2nd Stage	0	0	7	0	0	0	(batched)	47	293

P(CS) 1

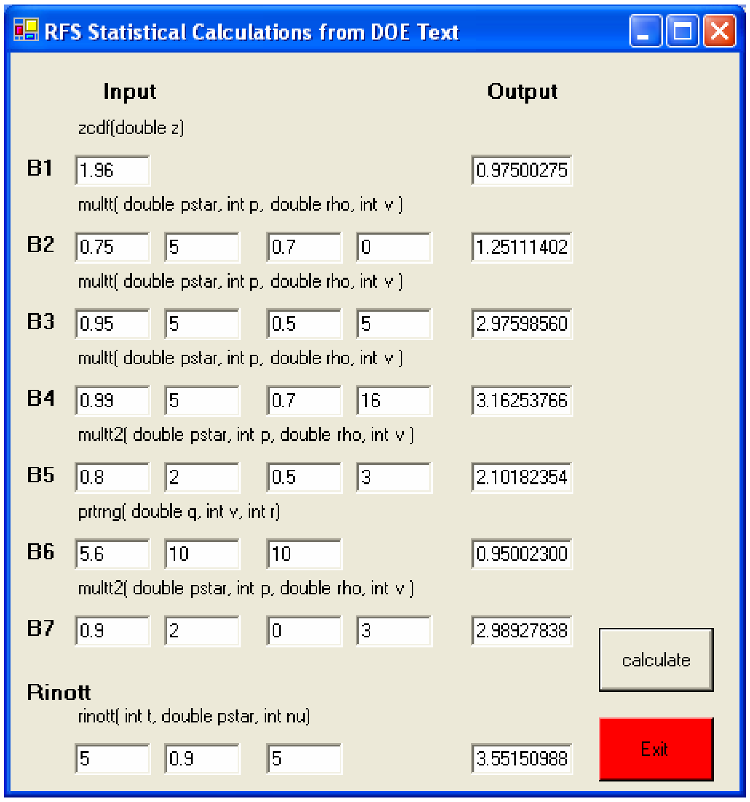
v1.007

Exit

Figure 34: Distributed Simulation BGP4 Module

APPENDIX B: RANKING AND SELECTION PARAMETER CALCULATION

Figure 35 presents sample test statistics necessitated by ranking and selection methods. Input and output columns follow from Bechhofer, Santner, and Goldsman (1995) table guidelines. Of interest, these test statistics are available at runtime. Test statistics include multivariate normal and t-distribution equicoordinate points with varying correlation inputs. Studentized range and maximum modulus parameters are also available. Historically, table lookups or FORTRAN code were used for these calculations.



RFS Statistical Calculations from DOE Text

	Input				Output
	zcdf(double z)				
B1	1.96				0.97500275
	mult(double pstar, int p, double rho, int v)				
B2	0.75	5	0.7	0	1.25111402
	mult(double pstar, int p, double rho, int v)				
B3	0.95	5	0.5	5	2.97598560
	mult(double pstar, int p, double rho, int v)				
B4	0.99	5	0.7	16	3.16253766
	mult2(double pstar, int p, double rho, int v)				
B5	0.8	2	0.5	3	2.10182354
	prtrng(double q, int v, int r)				
B6	5.6	10	10		0.95002300
	mult2(double pstar, int p, double rho, int v)				
B7	0.9	2	0	3	2.98927838
Rinott	rinott(int t, double pstar, int nu)				
	5	0.9	5		3.55150988

Figure 35: Ranking and Selection Parameter Calculations

REFERENCES

Banks, J., Carson, J. S., et al. (2001). *Discrete-Event System Simulation*. Upper Saddle River, New Jersey, Prentice Hall.

Bechhofer, R. E., (1954). A Single-Sample Multiple Decision Procedure for Ranking Means of Normal Populations with Known Variances. *Ann. Math. Sta.* 25, 16-39.

Bechhofer, R. E., Santner, T. J., and D. M. Goldsman. (1995). *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons.* New York, John Wiley & Sons, Inc.

Behforooz, A. and F. Hudson. (1996). *Software Engineering Fundamentals.* New York, Oxford University Press.

Bertsimas, D. and S. S. Patterson. (1998). The Air Traffic Flow Management Problem with Enroute Capacities. INFORMS *Operations Research,* 46(3):406-423, Linthicum, MD, Institute for Operations Research and the Management Sciences.

Blom, H. A. P., Klompstra, M. B., and B. Bakker. (2001). Accident Risk Assessment of Simultaneous Converging Instrument Approaches. In the *4th USA/Europe Air Traffic Management R&D Seminar*, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Brandt, K. and E. Roland. (1993). Modeling Coalition Warfare: A Multi-Sided Simulation Design. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G.W. Evans, M. Mollaghasemi, E.C. Russel, and W.E. Biles, 977-983, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Callaham, M. B. (1997). National Airspace System Architecture Metrics Assessment. *16th Digital Avionics Systems Conference,* AIAA/IEEE, 2:6.4-17.6.4-24, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Carothers, C. D. (2002). XSim: Real-Time Analytic Parallel Simulations. In *Proceedings of the 16th Workshop on Parallel and Distributed Simulation*, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Chen, T. L. (2000). *Design and Evaluation of an In-Cockpit Re-Planning Tool as an Emergency Decision Aid.* M.S. Thesis, Georgia Institute of Technology, Atlanta, Georgia.

Chick, S. E., and K. Inoue. (2001). New Procedures to Select the Best Simulated System Using Common Random Numbers. INFORMS *Management Science,* 47(8):1133-1149, Linthicum, MD, Institute for Operations Research and the Management Sciences.

Chin, D. K. and F. Melone. (1999). Using Airspace Simulation to Assess Environmental Improvements from Free Flight and CNS/ATM Enhancements. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P.A. Farrington, H.B. Nembhard, D. T. Sturrock, and G.W. Evans, 2:1295-1301, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Corker, K. M. (1999). Human Performance Simulation in the Analysis of Advanced Air Traffic Management. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, 821-829, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Diamond, P., Kloeden, P. E., Kozyakin, V. S., and A. V. Pokrovskii. (1997). A Model for Roundoff and Collapse in Computation of Chaotic Dynamical Systems. *Mathematics and Computers in Simulation*, 44:163-185, Orlando, FL, Elsevier.

Falker, J. M. and J. K. Kuchar. (2001). Analytical and Empirical Analysis of the Impacts of Restricting Airspace. In the *4th USA/Europe Air Traffic Management R&D Seminar*, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Fishman, G. (2001). *Discrete-Event Simulation: Modeling, Programming, and Analysis*. New York, Springer-Verlag.

Fishwick, P. A. and R. Modjeski (1991). *Knowledge-Based Simulation: Methodology and Application (Advances in Simulation Vol.4)*. New York, Springer-Verlag.

Fondacci, R., Goldschmidt, O., and V. Letrouit. (1998). Combinatorial Issues in Air Traffic Optimization. INFORMS *Transportation Science*, 32(3):256-268 Linthicum, MD, Institute for Operations Research and the Management Sciences.

Fujimoto, R. M. (2000). *Parallel and Distributed Simulation Systems.* New York, John Wiley & Sons, Inc.

Fujimoto, R. M. (2001). Parallel and Distributed Simulation Systems. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, 147-157, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Frolow, I. and J. H. Sinnott. (1989). National Airspace System Demand and Capacity Modeling. In *Proceedings of the IEEE* 77(11):174-186, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Goldsman, D., Kim, S., Marshall, W. S., and B. L. Nelson. (2002). Ranking and Selection for Steady-State Simulation: Procedures and Perspectives. INFORMS *Journal on Computing*, 14 (1):2-19, Linthicum, MD, Institute for Operations Research and the Management Sciences.

Glynn P. W. and D. L. Iglehart. (1990). Simulation Output Analysis Using Standardized Time Series. *Mathematics of Operations Research*, 15:1-16.

Glynn P. W. and W. Whitt. (1990). Estimating the Asymptotic Variance with Batch Means. *Operations Research Letters,* 10:431-435.

Hayes, C. C. (1999). Agents in a Nutshell – a Very Brief Introduction. *IEEE Transactions on Knowledge and Data Engineering* 11 (1):127-132, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Hatley, D. J., and I. A. Pirbhai. (1988) *Strategies for Real-Time System Specification.* New York, Dorset House Publishing Co., Inc.

Holden, T. C. and F. Wieland. (2003). Runway Schedule Determination by Simulation Optimization. In *Proceedings of the 2003 Winter Simulation Conference,* ed. S. Chick, P.J. Sanchez, D. Ferrin, and D.J. Morrice, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Hopp, W. J., and M. I. Spearman. (2000) *Factory Physics.* 2nd Ed. New York, McGraw-Hill.

Karatza, H. D. and R. C. Hilzer. (2002). Load Sharing in Heterogeneous Distributed Systems. In *Proceedings of the 2002 Winter Simulation Conference,* ed. E. Yucesan, C.H. Chen, J.L. Snowdon, and J.M. Charnes, 489-496, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Kim, S. and B. Nelson. (2001). A Fully Sequential Procedure for Indifference-Zone Selection in Simulation. *ACM TOMACS*, 11:251-273.

Kettenis, D. L. (1997). An Algorithm for Parallel Combined Continuous and Discrete-Event Simulation. In *Simulation Practice and Theory,* 5:167-184, Orlando, FL, Elsevier.

Kleinman, N. L., Hill, S. D., and V. A. Ilenda. (1998). Simulation Optimization of Air Traffic Delay Cost. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, 1177-1181, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Kochan, S. G. (1994). *Programming in ANSI C.* Revised Edition. Indianapolis, Indiana, SAMS – Prentice Hall.

Kostiuk, P. F. (2001). Demand Management versus Capacity Enhancement: Which Direction for Air Transportation? In the *4th USA/Europe Air Traffic Management R&D Seminar*, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Knorr, D., Post, J., Walker, M., and D. Howell. (2001) An Operational Assessment of Terminal and En Route Free Flight Capabilities. In the *4th USA/Europe Air Traffic Management R&D Seminar*, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Law, A. M. and W. D. Kelton. (2000). *Simulation Modeling and Analysis*. 3rd Edition. Boston, MA, McGraw Hill.

Lee, S. M., A. R. Pritchett, and D. Goldsman. (2001). Hybrid Agent-Based Simulation for Analyzing the National Airspace System. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, 1029-1036, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Lee, S.M. (2002). *Agent-Based Simulation of Socio-Technical Systems: Software Architecture and Timing Mechanisms.* Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, Georgia.

Leveson, N. et al. (2001). A Safety and Human-Centered Approach to Developing New Air Traffic Management Tools. In the *4th USA/Europe Air Traffic Management R&D Seminar*, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Logan, B. and G. Theodoropoulos. (2001). Multi-Agent Systems and Agent-Based Simulation. In *Proceedings of the IEEE* 29(2):174-186, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Meketon, M. S. and B. W. Schmeiser. (1984). Overlapping Batch Means: Something for Nothing? In *Proceedings of the 1984 Winter Simulation Conference*, ed. S. Sheppard, U. W. Pooch, and C. D. Pegden, 227-230, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

National Research Council (NRC) (2002). *Modeling and Simulation in Manufacturing and Defense Systems Acquisition*. Washington D.C., National Academy of Sciences.

National Airspace System (NAS) Report Card. (1999). *MITRE Corporation*. McLean, Virginia.

Nelson, B. L., and D. Goldsman. (2001). Comparisons with a Standard in Simulation Experiments. INFORMS *Management Science*, 47(3):449-463, Linthicum, MD, Institute for Operations Research and the Management Sciences.

Neter, J., Kutner, M. H., Nachtsheim, C. J., and W. Wasserman. (2001). *Applied Linear Statistical Models 4th ed.* New York, McGraw-Hill.

Oliver, D. W.. (1997). Engineering of Complex Systems with Models. In *IEEE Transactions on Aerospace and Electronic Systems*, 33(2):667-685, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Odoni, A. R. (1991). Issues in Modeling a National Network of Airports In *Proceedings of the 1991 Winter Simulation Conference*, ed. B.L. Nelson, W.D. Kelton, G.M. Clark, 756-762, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Pallottino, L. and A. Bicchi. (2000). On the Optimal Conflict Resolution for Air Traffic Control. In *Proceedings of Intelligent Transportation Systems,* Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Pillage, L. T., Rohrer, R. A., and C. Visweswariah. (1995). *Electronic Circuit and System Simulation Methods.* New York, McGraw-Hill, Inc.

Pinedo, M. (1995). *Scheduling. Theory, Algorithms, and Systems.* Englewood Cliffs, New Jersey. Prentice Hall.

Pritchett, A. R., Lee, S., Abkin, M., Gilgur, A. Z., Bea, R. C., Corker, K. M., Verma, S., and A. Jadhav. (2002). Examining Air Transportation Safety Issues Through Agent-Based Simulation Incorporating Human Performance Models. In

Proceedings of 2002 Digital Avionics Systems Conference, 2:27-31, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Pritchett, A. R. and C. Ippolito. (2000). Software Architecture for a Reconfigurable Flight Simulator. In *Proceedings of the AIAA Modeling and Simulation Technologies Conference,* Denver, CO.

Reiss, R. D. and M. Thomas (2001). *Statistical Analysis of Extreme Values. 2nd ed.* Basel, Birkhauser Verlag.

Rinott, Y. (1978). On Two-Stage Selection Procedures and Related Probability-Inequalities. *Commun. Stat. –Theory and Methods* A8, 799-811.

Ross, S. M. (1996). *Stochastic Processes. 2nd ed.* New York, John Wiley & Sons.

Rolfe, J. M. and K. J. Staples. (1986). *Flight Simulation.* Melbourne, Australia, Cambridge University Press.

Sanchez, S. M and T. W. Lucas. (2002). Exploring the World of Agent-Based Simulations: Simple Models, Complex Analyses. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yucesan, C.H. Chen, J.L. Snowdon, and J.M. Charnes, 116-126, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Schaefer, A. J., Goldsman, D., Johnson, E., Kleywegt, A. J., and G. L. Nemhauser. (2002). A Stochastic Model of Airline Operations. INFORMS *Transportation Science,* 36(4):357-377, Linthicum, MD, Institute for Operations Research and the Management Sciences.

Schwabacher, M., and A. Gelsey. (1998). Multilevel Simulation and Numerical Optimization of Complex Engineering Designs. In *Journal of Aircraft,* 35(3):387-397. Reston, VA, American Institute of Aeronautics and Astronautics, Inc.

Schwartz, J., Mundra, A., Broderick, J., and R. Nash (1997). Some ATC Implications of Introducing Flight Management System Based Routes in the Terminal Airspace. *16th Digital Avionics Systems Conference,* AIAA/IEEE, 2:9.1-16-9.1-23, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Seber, G.A.F. (1997). *Linear Regression Analysis.* New York, John Wiley & Sons.

Tan, G., Ng, W. N., and F. Moradi. (2001). An Enroute ATC Simulation Experiment for Sector Capacity Estimation. In *IEEE Transactions on Control Systems Technology*, 1:(3)138-144, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Tofukuji, N. (1993). Engineering of Complex Systems with Models. In *IEEE Transactions on Aerospace and Electronic Systems*, 33(2):667-685, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Wieland, F. (1998). Parallel Simulation for Aviation Applications. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D.J. Mediros, E.F. Watson, J.S. Carson, and M.S. Manivannan, 1191-1199, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Wilson, J. R. (2001). A Multiplicative Decomposition Property of the Screening-and-Selection Procedures of Nelson Et Al. INFORMS *Operations Research*, 49(6):964-966, Linthicum, MD, Institute for Operations Research and the Management Sciences.

Wilson, I. and K. Fleming (2002). Controller Reactions to Free Flight in a Complex Transition Sector Re-Visited Using ADS-B+. *21st Digital Avionics Systems Conference*, AIAA/IEEE, 1:2.B.5-1-2.B.5-10, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Zeitlin, Andrew D. (2001). Safety Assessments of ADS-B and ASAS. In the *4th USA/Europe Air Traffic Management R&D Seminar*, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

Zeghal, K. and E. Hoffman (2000). Design of cockpit displays for limited delegation of separation assurance. *18th Digital Avionics Systems Conference*, AIAA/IEEE, D:2-1-D2-8, Piscataway, New Jersey, Institute of Electrical and Electronics Engineers.

VITA

Kirk C. Benson

Kirk C. Benson was born in Renton, Washington on March 28, 1964. He received a B.S. in Mechanical Engineering from the United States Military Academy in 1986 and obtained a commission in the U.S. Army as an engineer. After serving in Germany and Hawaii in positions ranging from platoon leader to company commander, he obtained a M.S. in Operations Research from the Naval Postgraduate School in 1997. His master thesis research involved modeling data encapsulation and a communications network for the National Training Center located at Fort Irwin, California. Subsequently, he served as an assistant professor in the Department of Mathematical Sciences at the United States Military Academy. In 2001, he entered the doctoral program in the School of Industrial and Systems Engineering at the Georgia Institute of Technology in Atlanta, Georgia. His minor is in Adaptive System Control Methods.

He is married to Shelley and enjoys a variety of pursuits to include weightlifting, skiing, and reading.