Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

EFFICIENT SIMULATION OF THE NATIONAL AIRSPACE SYSTEM

John F. Shortle
Donald Gross

Systems Engineering & Operations Research
George Mason University
Fairfax, VA 22030, U.S.A.

Brian L. Mark

Electrical & Computer Engineering
George Mason University
Fairfax, VA 22030, U.S.A.

ABSTRACT

The National Airspace System (NAS) is a large and complicated system. Detailed simulation models of the NAS are generally quite slow, so it can be difficult to obtain statistically valid samples from such models. This paper presents two methods for reducing the complexity of such networks to improve simulation time. One method is removal of low-utilization queues – that is, replacing a queueing node with a delay node, so that airplanes experience a service time at the node but no queueing time. The other is removal of nodes by clustering – that is, where groups of nodes are collapsed into a single node. We employ the methods on simple networks and show that the reductions yield very little loss in modeling accuracy. We provide some estimates for the potential speedup in simulation time when using the methods on large networks.

1 INTRODUCTION

The National Airspace System (NAS) is a large and complicated system. The NAS contains about 30 large hubs (like Atlanta) about 30 medium hubs (like Cleveland), about 50 small hubs (like Colorado Springs), about 600 more airports with scheduled flights, and thousands of other public-use airports. For example, Figure 1 shows all public-use airports within 1,000 nautical miles (Nm) of Virginia that have runways over 3,000 feet. Operations at the hub airports can be quite complex, since airplanes must traverse through arrival sectors, to the runway, through a series of taxi-ways, to a gate, and then back out.

The NAS also contains about 500 airspace sectors at varying altitude levels. On a given day, there can be about 5,000 commercial flights in the air at one time (not including general aviation planes, military planes, etc). The interaction between these airplanes can be complex, as controllers employ a range of tactics to safely separate and sequence the airplanes.

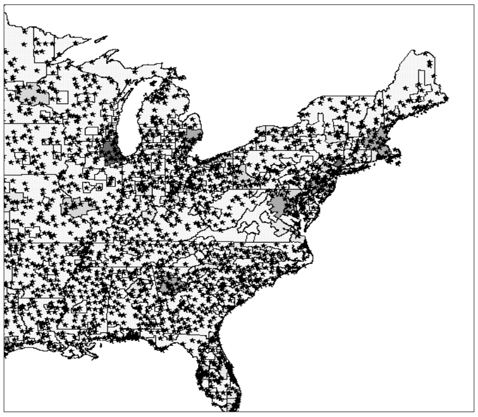
Figure 1: Airports within 1,000 Nm of Virginia

Thus, highly detailed simulation models of the NAS tend to be quite slow. This makes it difficult to obtain statistically valid samples from such models. One example is the Total Airspace and Airport Modeler (TAAM). The model simulates the physical positions of individual airplanes throughout the network. The model has been widely used to study traffic at single, large airports, but it can also be used to simulate the whole NAS. For example, Yousefi (2003) created an implementation of TAAM to simulate all flights – including general aviation flights – in the northeast corridor. The model took about 8 hours of real time to generate 24 hours of simulated time. Since this is only one replication, obtaining a statistically valid sample via multiple replications would be very time consuming.

On the other end of the spectrum are much simplified models. One example is LMINET (Gaier and Kostiuk 1998) which uses analytical queueing models to reduce the simulation time of the whole network. This model is fast, but relies on some questionable assumptions, such as Poisson arrivals to each node of the network. Another example is DPAT (MacReynolds and Sinnot 1998) which also uses a simplified queueing network representation, but has a very limited Monte Carlo simulation capability.

The purpose of this paper is to investigate ways of creating a simulation model which is both fast and reasonably accurate. Naturally, there is an inherent trade-off between speed and accuracy, so we seek a model that is in-between LMINET and TAAM, but hopefully closer to LMINET in speed and closer to TAAM in accuracy.

Much recent work on large-network simulations has been motivated by Internet applications. Here, network size is even more problematic than for the NAS. For example, Riley and Ammar (2002) conservatively estimate that it would take *a year* of CPU time to simulate *100 seconds* of activity on the Internet. Techniques for simulating large Internet networks generally fall into two categories: Fluid simulations (e.g., Y. Liu et al. 2003, B. Liu et al. 1999) and parallel simulations (e.g., Cowie et al. 1999, Rao and Wilsey 1999). For a survey, see Riley and Ammar (2002).

Since fluid models would not provide enough fidelity to model airplane conflicts and interactions, this paper focuses on other techniques for improving simulation speed. Specifically, we examine two types of *network reductions* as a means of improving simulation time: Elimination of the queueing time at low-utilization nodes (Section 3) and clustering of multiple nodes into a single node (Section 4). We also estimate the potential benefit in applying these methods to a full NAS simulation.

2 MODELING FRAMEWORK

In general, when simulating the NAS, we are not interested in the performance of the network at *every node*. Rather, we tend to be interested in performance metrics at specific *subsets* of nodes. For example, if we want to know the effect of adding a runway at Washington Dulles Airport (IAD), we primarily care about delays at IAD and nearby airports. We also may be interested in delays at other major airports in the NAS, particularly those with many direct flights to and from IAD. Our approach is to divide the entire network into two sub-networks: **C** and **R**. The subnetwork **C** contains the core nodes of interest, and the subnetwork **R** contains the remaining nodes.

As an example, consider a network consisting of the 12 airports in Table 1. In this sample network, **C** consists of 4 large, congested airports, and **R** consists of 8 medium airports. Suppose we are directly interested in the performance of the large airports. We are not *directly* interested in the nodes in **R**, except that they might have an impact on the nodes in **C**. (The table also gives the average number of daily commercial flights leaving each airport during August, 2001, before 9/11/01. Although we have applied real names to the nodes in the network, it is only a simple example and not intended to represent a real network.)

Now, the goal is to create a new network that is more efficient to simulate than the original network, but still gives accurate estimates for the performance metrics of **C**. In particular, we seek a smaller, surrogate subnetwork $\mathbf{R}^*$ to replace **R**, such that simulating **C** with $\mathbf{R}^*$ gives essentially the same performance metrics for **C** as simulating **C** with **R**.

Table 1: Example Network of 12 Nodes

	Airport	Code	Daily Flights
C	Atlanta	ATL	1,157
	Chicago	ORD	1,245
	Newark	EWR	583
	San Francisco	SFO	467
R	Albuquerque	ABQ	140
	Cleveland	CLE	364
	Indianapolis	IND	270
	Kansas City	MCI	280
	Memphis	MEM	438
	New Orleans	MSY	179
	Raleigh-Durham	RDU	288
	San Jose	SJC	246

In this paper, we use two techniques to reduce **R** to a smaller network $\mathbf{R}^*$, discussed in the next two sections:

1. Removal of low-utilization queues. By this, we mean replacing a queueing node with a delay node, so that an arriving airplane has a service time at the node but no queueing time.
2. Node clustering.

3 LOW-UTILIZATION NODES

In this section, we discuss simplifying **R** by removing low-utilization queues (that is, by removing the queueing time at a node, but retaining the service time). We motivate this technique with the following example. Figure 2 shows the theoretical waiting time of an M/M/1 queue with service rate $\mu = 10$. For arrival rates λ below about 6, the expected waiting time curve is nearly flat. For example, if this queue represents an airport and $\lambda = 3$ on a "typical" day while $\lambda = 5$ on a "bad" day (due to re-routing from other airports), the queueing is essentially the same for either case. In fact, we could probably eliminate the queueing time (that is, by replacing the M/M/1 queue with an M/M/∞ queue, or in simulation, by replacing the queueing node with a delay node) without impacting the rest of the network too much.

3.1 Jackson Networks

We test the plausibility of this approach with a Jackson network, since we can solve such networks analytically. Our test network contains the 12 airports in Table 1. Figure 3 shows the basic network structure using only 3 airports (the full 12-airport network is similar). Each airport consists of 2 nodes: an arrival node and a gate node. The arrival node models the sequencing and queueing of airplanes as they approach the runway (for simplicity, we do not include an analogous *departure* node, but the example could easily

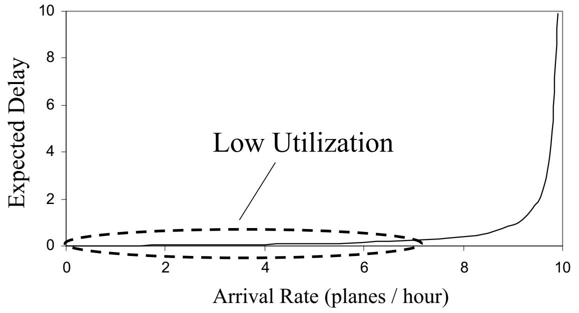

Figure 2: Waiting Time of M/M/1 Queue ($\mu = 10$)

Table 2: Jackson Network Parameters

	i	Airport	Arrival Node $\mu_{A,i}$	Arrival Node $S_{A,i}$	Gate Node $\mu_{G,i}$	Gate Node $S_{G,i}$
C	1	ATL	30	1	1	99
	2	ORD	35	1	1	99
	3	EWR	30	1	1	99
	4	SFO	40	1	1	99
R	5	ABQ	20	1	1	99
	6	CLE	20	1	1	99
	7	IND	20	1	1	99
	8	MCI	20	1	1	99
	9	MEM	20	1	1	99
	10	MSY	20	1	1	99
	11	RDU	20	1	1	99
	12	SJC	20	1	1	99

be generalized in this way). There is one server (i.e., the runway) with a service rate between 20 and 40 landings per hour (Table 2). The gate node models the turnaround times for the airplanes. We assume a large number of gates (here 99) so there is little or no queueing. For notation, $\mu_{A,i}$ and $\mu_{G,i}$ are the service rates for the arrival and gate nodes of airport i (service times are i.i.d. exponential), and $S_{A,i}$ and $S_{G,i}$ are the number of servers at the respective nodes. Once an airplane leaves node i, it goes to node j with probability p_{ij}. No airplanes enter or exit the system. We choose the transition probabilities p_{ij} (not shown in Table 2) to be proportional to observed flights between the airports during the month of August, 2001.

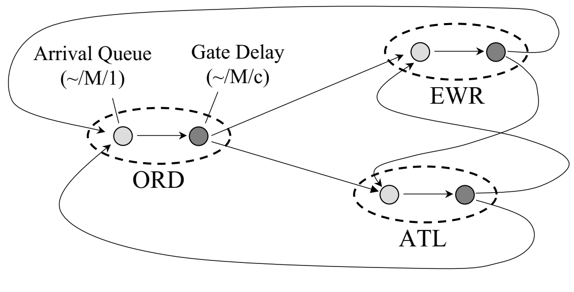

Figure 3: Basic Network Structure

We can solve this network analytically (e.g., using the QTS package in Gross and Harris 1998). Table 3 shows the results, using 150 airplanes in the closed network. $E(W_q)$ is the average time an airplane spends *in the queue* at a given node; $E(W)$ is the average *total* time an airplane spends at a node (queue time plus service time). We only show statistics for the arrival nodes, since the gate nodes have essentially zero queueing. (The congestion levels do not represent actual congestion levels at these airports.)

Table 3: Baseline Jackson Network

	Node	ρ	$E(W)$	$E(W_q)$
C	ATL	0.800	0.1519	0.1185
	ORD	0.797	0.1289	0.1004
	EWR	0.583	0.0785	0.0452
	SFO	0.200	0.0312	0.0062
R	ABQ	0.075	0.0540	0.0040
	CLE	0.477	0.0947	0.0447
	IND	0.441	0.0887	0.0387
	MCI	0.408	0.0839	0.0339
	MEM	0.635	0.1335	0.0835
	MSY	0.287	0.0699	0.0199
	RDU	0.408	0.0839	0.0339
	SJC	0.184	0.0612	0.0112

Now, we take the sub-network **R** and replace it with a simpler network **R*** to see how the performance metrics for **C** change. We do this by eliminating queues in **R** which have utilizations below a given level. For analytical solutions of Jackson networks, we effectively eliminate a queue by changing the number of servers at a given node to some large value, say 99. For simulation (next subsection), we replace the queueing node with a delay node. We test the following cases:

0. Baseline (no change: **R*** = **R**).
1. Eliminate *all queues* in **R**. (For $i = 5, \cdots, 12$, set $S_{A,i} = 99$ in **R***.)
2. Eliminate all queues in **R** with $\rho < 0.45$. (For $i = 5, 7, 8, 10, 11, 12$, set $S_{A,i} = 99$ in **R***.)
3. Eliminate all queues in **R** with $\rho < 0.4$. (For $i = 5, 10, 12$, set $S_{A,i} = 99$ in **R***.)
4. Eliminate all queues in **R** with $\rho < 0.01$. (Only the gate nodes have utilizations less than 0.01. Since these nodes already have 99 servers, this yields no change to the *analytical* Jackson-network solution. However, it does make a difference in speed for the *simulated* solution, since this involves changing queueing nodes to delay nodes.)
5. Repeat cases 1-4 using an *adjusted* service rate for all queues that have been eliminated. To show how this works, consider the Albuquerque airport (ABQ). In the baseline case, the average *queue* time W_q at the arrival node is 0.0040 (Table 3). The average *total* time W at the node is 0.0540. If we lump the queue and the service times together

and suppose this is all part of the service time, then the adjusted service rate for this queue is $\mu' = 1/E(W) = 18.519$. For case 3b, for example, we make the following changes to $\mathbf{R}$ to get $\mathbf{R}^*$: $S_{A,5} = S_{A,10} = S_{A,12} = 99$; $\mu_5 = 18.519$, $\mu_{10} = 14.306$, $\mu_{12} = 16.340$. Cases 1b and 2b are similar.

Table 4 shows the performance of these simplified networks, compared to the baseline. The table shows the percent difference in the expected queue waiting time $E(W_q)$ at the 4 arrival nodes in $\mathbf{C}$ (all values are rounded to the nearest 0.01%). In most cases, there is little change in $E(W_q)$ from the original network to the simplified network. Only in case 1, where we eliminated *every* queue in $\mathbf{R}$, does $E(W_q)$ change by more than 3%. Naturally, as we eliminate fewer queues (going from case 1 to 4), the accuracy improves; conversely, the time to simulate these networks increases.

Table 4: Performance of Simplified Networks

Case	ATL	ORD	EWR	SFO
Baseline	–	–	–	–
1	6.30%	6.24%	3.40%	1.82%
2	2.51%	2.49%	1.37%	0.74%
3	0.38%	0.37%	0.21%	0.11%
4	0.00%	0.00%	0.00%	0.00%
1b	-0.26%	-0.25%	-0.07%	-0.01%
2b	-0.08%	-0.08%	-0.02%	0.00%
3b	-0.01%	-0.01%	0.00%	0.00%
4b	0.00%	0.00%	0.00%	0.00%

Using the *adjusted* service times (cases 1b-4b) greatly increases the accuracy of the reduced network (for example, compare cases 1b-4b with cases 1-4). However, to get these values, we must solve the original network, thus defeating the purpose of using the simpler network. Nevertheless, this approach may still be useful. In collecting data to populate a model, it is often easier to estimate the total time at a node (which is precisely the adjusted service time used here), rather than to estimate the component "service" and "queueing" times. This is particularly true for airspace sectors. So, for low-utilization queues, it may be possible to estimate these values from actual data.

The performance of the simplified networks are quite good. One explanation comes from considering the *open* version of the corresponding Jackson network. In an open network, we can determine the performance metrics at each node by solving (e.g., Gross and Harris 1998)

$$\vec{\lambda} = \vec{\gamma} + \mathbf{P}\vec{\lambda}$$

for $\vec{\lambda}$, where $\vec{\lambda}$ is the vector of *total* arrival rates to each node, $\vec{\gamma}$ is the vector of arrival rates to each node *from the outside* and $\mathbf{P}$ is the routing matrix. This equation does not depend on the number of servers S_i or the service rate μ_i at any node. Therefore, changes to S_i or μ_i do not change the solution for $\vec{\lambda}$.

In particular, the previous reductions of $\mathbf{R}$ to $\mathbf{R}^*$ involved changing the number of servers S_i at some nodes from a finite value to an infinite value, but kept the routing matrix $\mathbf{P}$ fixed. In an *open* network, such changes would not affect the solution $\vec{\lambda}$. Since the performance metrics of node i depend on λ_i, μ_i, and S_i, changing S_j or μ_j at a *different* node j does not change λ_i and therefore has *no* effect on the metrics at node i ($j \neq i$). In other words, in an open network, the previous reductions of $\mathbf{R}$ to $\mathbf{R}^*$ have *no* affect on the performance metrics observed in $\mathbf{C}$. Thus, it is not surprising that the effects of analogous reductions in closed networks are small.

3.2 Simulation

We now investigate the potential simulation benefit in removing low-utilization queues. That is, we investigate how much simulation time is saved by changing a queueing node to a delay node. To do this, we simulate the 24-node (12-airport) closed Jackson network described previously in the Arena simulation package (version 5). We compare the baseline network to the 4 network reductions. Previously, to remove a queue at node i, we changed the number of servers to a large value. In simulation, we remove the queue by replacing the queueing node with a *delay* node. For example, in case 3, we change *arrival* nodes 5, 10, and 12 to delay nodes and we change all twelve *gate* nodes to delay nodes. In case 4, we change the twelve gate nodes to delay nodes.

Table 5: Simulation Speedup

	Time	W_q Error			
Case	Reduction	ATL	ORD	EWR	SFO
Baseline	–	-1.1%	0.2%	1.0%	-0.4%
1b	31%	-0.9%	-0.2%	0.9%	-0.6%
2b	27%	-0.6%	0.7%	0.1%	0.5%
3b	21%	-0.2%	0.4%	0.7%	0.5%
4b	19%	-1.1%	0.2%	1.0%	-0.4%

Table 5 shows the improvements in speed for cases 1b-4b (the simulation times for cases 1-4 and cases 1b-4b, respectively, are essentially the same.) Case 1b, which is the smallest network, has the greatest time reduction (31%). The table also shows the percent errors of the queue waiting times compared to theoretical values. In general, the simulation randomness dominates the network-reduction errors.

We now establish an approximate upper bound for the reduction in simulation time. In an "ideal" case, $\mathbf{C}$ and $\mathbf{R}$ are two independent networks. Then, we can simulate $\mathbf{C}$ by itself and ignore $\mathbf{R}$. Now, in our baseline 12-airport network, about 29% of all operations occurred in $\mathbf{C}$ and about 71% occurred in $\mathbf{R}$. Thus, at best, letting $\mathbf{R}^*$ be the empty set gives a reduction in simulation time of 71%. In Table 5, by eliminating low-utilization queues (for example, Case 2), we were able to achieve almost 40% of this “ideal" reduction.

4 NODE CLUSTERING

In this section, we investigate *node clustering* as a way of reducing the subnetwork $\mathbf{R}$ to a simpler subnetwork $\mathbf{R}^*$. The basic idea is to group sets of nodes in $\mathbf{R}$ into clusters and then to represent each cluster as a single node, thereby obtaining $\mathbf{R}^*$. Two critical questions are:

1. Which nodes should be grouped into clusters?
2. How can a cluster be replaced by a single node without sacrificing modeling accuracy?

Norton’s theorem (Chandy et al. 1975) provides precise answers to these questions *for Jackson networks*. Specifically, the theorem specifies conditions under which a set of nodes can be clustered together as a single node such that the resulting network is equivalent to the original network.

In this section, we review Norton’s theorem and its extensions. Then, we apply Norton-type reductions to air traffic networks, and we estimate the potential benefit in simulation speed.

4.1 Norton’s Theorem

Consider a closed Jackson network with M nodes and N customers (or airplanes). Suppose we are interested only in the queueing behavior at node M. That is, $\mathbf{C} = \{M\}$ and $\mathbf{R} = \{1, \cdots, M-1\}$.

Norton’s theorem states that we can replace $\mathbf{R}$ with a single node R_{eq} having a state-dependent service rate $\mu(n)$ as follows: Let $\lambda^*_M(n)$ be the throughput at node M in a modified network obtained by setting the service rate at node M to infinity (effectively removing node M) and setting the network population to n. Then, we can reduce the M-node network to an equivalent two-node network consisting of node M and a new node R_{eq} with state-dependent service rate $\mu(\cdot)$ satisfying:

$$\mu(n) = \lambda_M(n).$$

(That is, when there are n customers at the new node, the service rate at this node is $\mu(n)$.)

A more general form of Norton’s theorem (Boucherie and van Dijk 1993) applies when $\mathbf{C}$ consists of more than one node. In this case, we can reduce a cluster of nodes in $\mathbf{R}$ to a single node provided the cluster contains a single node of entry (or *ingress*) and a single node of exit (or *egress*). We call such clusters *Norton-type* clusters. A cluster of one node is trivially a Norton-type cluster. Thus, we can partition $\mathbf{R}$ into l Norton-type clusters $\mathbf{R}_1, \cdots, \mathbf{R}_l$, each of which satisfies this property (where some “clusters" may be only single nodes). Applying the Norton procedure (below), we can reduce each cluster $\mathbf{R}_i$ to an equivalent node $R_{eq,i}$, so the surrogate network is $\mathbf{R}^* = \{R_{eq,1}, \cdots, R_{eq,l}\}$.

The basic steps for replacing a Norton-type cluster $\mathbf{R}_i$ with an equivalent node $R_{eq,i}$ are:

1. Consider the cluster $\mathbf{R}_i$ as a network in isolation with a “short-circuit” path directed from the egress of the cluster to the ingress. We call the resulting network the *short-circuited* representation of the cluster.
2. For the short-circuited network, find the state-dependent throughput $\lambda(n)$ along the short-circuit path from ingress to egress, where n is the number of customers in the short-circuited network, $n = 1, \cdots, N$. For Jackson networks, this can be done by applying the state-dependent Buzen algorithm (Gross and Harris 1998 give the state-*independent* Buzen algorithm, which can easily be generalized to the state-dependent case).
3. Replace the cluster $\mathbf{R}_i$ with a single node $R_{eq,i}$ with state-dependent service rate $\mu(n) = \lambda(n)$.

4.2 Example

Figure 4 shows an example network containing three airports. The airports at Newark and Atlanta are modeled as single nodes, while the airport at Chicago is modeled as a four-node subnetwork. The Chicago subnetwork consists of two arrival nodes (1 and 2), representing the delays that occur when planes wait to land and taxi, and two departure nodes (3 and 4), representing the delays that occur when planes wait for departure. The numbers associated with the arcs represent the routing probabilities for the network. The two dark circles within the Chicago subnetwork represent the ingress and egress of the cluster, and we assume that the service times at these two nodes are instantaneous. Effectively, the total number of nodes in the network is $M = 6$ and the routing matrix is

$$Q = \begin{bmatrix} 0 & 0 & 0.9 & 0.1 & 0 & 0 \\ 0 & 0 & 0.2 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 0.6 \\ 0 & 0 & 0 & 0 & 0.4 & 0.6 \\ 0.56 & 0.24 & 0 & 0 & 0 & 0.2 \\ 0.49 & 0.21 & 0 & 0 & 0.3 & 0 \end{bmatrix}.$$

To make this example concrete, we further assume that the nodes have constant service rates:

$$\begin{aligned} &\mu_1 = 30,\ \ \mu_2 = 35,\ \ \mu_3 = 40, \\ &\mu_4 = 45,\ \ \mu_5 = 30,\ \ \mu_6 = 30, \end{aligned} \tag{1}$$

and that the number of planes (i.e., customers) is $N = 70$. The core set of nodes is $\mathbf{C} = \{5, 6\}$ and the remaining set

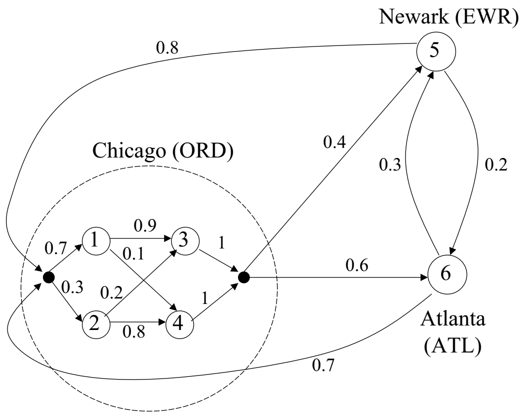

Figure 4: Example Air Traffic Network

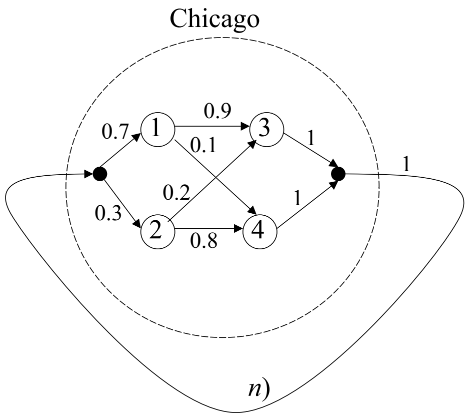

Figure 5: Short-Circuited Representation of Chicago Airport

of nodes is $\mathbf{R} = \{1, 2, 3, 4\}$ (the Chicago cluster). Our objective is to reduce $\mathbf{R}$ to a single node R_{eq} via a Norton-type reduction.

First, we create a short-circuited representation of the Chicago cluster as shown in Figure 5. The routing matrix for this network is

$$Q_{ch} = \begin{bmatrix} 0 & 0 & 0.9 & 0.1 \\ 0 & 0 & 0.2 & 0.8 \\ 0.7 & 0.3 & 0 & 0 \\ 0.7 & 0.3 & 0 & 0 \end{bmatrix}.$$

The service rates for this network are (same as (1))

$$\mu_1 = 30,\ \mu_2 = 35,\ \mu_3 = 40,\ \mu_4 = 45.$$

Applying the state-dependent Buzen algorithm to the network of Figure 5 yields the throughputs at all nodes of the network. In particular, the state-dependent throughput $\lambda(n)$ on the short-circuit path equals the sum of the throughputs at nodes 3 and 4 (i.e., $\lambda(n) = \lambda_3(n) + \lambda_4(n)$). Table 6 gives $\lambda(n)$ up to $n = 5$ (the complete table goes up to $n = N = 70$).

Table 6: State-Dependent Service Rates (ORD)

n	$\lambda(n)$
1	17.84
2	27.32
3	32.78
4	36.13
5	38.25
⋮	⋮

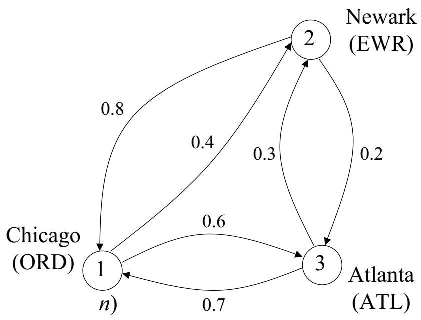

Figure 6: Reduced Air Traffic Network

Figure 6 shows the reduced network. Here, we represent the Chicago airport as a single node with state-dependent service rate $\mu_1(n)$, where

$$\mu_1(n) = \lambda(n).$$

The routing matrix of the reduced network is

$$Q_{red} = \begin{bmatrix} 0 & 0.4 & 0.6 \\ 0.8 & 0 & 0.2 \\ 0.7 & 0.3 & 0 \end{bmatrix}.$$

By Norton's theorem, the reduced network of Figure 6 is equivalent to the original network of Figure 4 with respect to the airports at Newark and Atlanta. We have simulated both Figure 4 and Figure 6 using Arena and have found that the time to simulate the reduced network is about 25% less than the original.

4.3 Application to Air-Traffic Networks

Intuitively, the basic criterion for Norton's theorem to work is that the cluster of nodes must effectively have a single point of entry and a single point of exit. One place where this occurs is at an airport. If an airport has only one runway, then that runway is a single point of entry and exit for the airport. This is true even if the airport model is quite complicated, consisting of many nodes for the runway, taxi-ways, gates, and so forth.

Airports with multiple runways are also potential candidates, though this is an area for further research. There are two basic cases. First, if there are two *dependent* runways (meaning that arrivals to one runway impact the flow of arrivals to the other runway), then the arrival *airspace* is a single node of entry, since permission to land on either runway depends on the utilization of the common arrival airspace. Second, if the runways are *independent*, then it may be possible to model the airport as two separate halves, where a Norton-type reduction is applied to each half. For example, at Atlanta, the arrival runways are on opposite sides of a large terminal, essentially splitting the airport into two. An open question is how well Norton-type reductions can be applied to such airports.

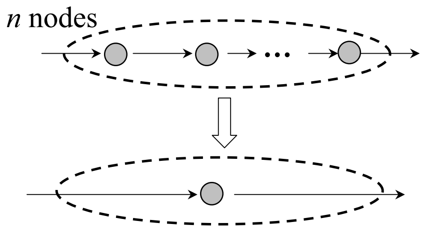

Figure 7: Reduction of Multi-Nodes to Single Node

To estimate the potential benefit of clustering in simulation, we consider an airport modeled with n nodes in series (Figure 7), and its corresponding reduction to a single node. The series model could represent, for example, the sequence of flying through the arrival airspace, landing on the runway, taxiing in, arriving at the gate, taxiing out, taking off, and finally leaving the departure airspace. In our simulations, we let the arrival rate be $\lambda = 0.5n$ and the service rate at each node be $\mu = n$. Thus, the utilization at each node is $\rho = 0.5$, and the expected *total* time at the airport is 1, regardless of n. The equivalent 1-node network is found using the Norton-type reduction, described previously. Simulating the 1-node network requires simulating a state-dependent service rate $\mu_1(\cdot)$, whereas in the n-node network, the service rates μ are all constant.

Figure 8 shows the time savings as a function of n. That is, let $T(n)$ be the time to simulate the n-node network, and let $T_1(n)$ be the time to simulate the equivalent single-node network (using Norton's theorem and state-dependent service). The y-axis in Figure 8 is

$$\text{\% Reduction in Simulation Time} = \frac{T(n) - T_1(n)}{T(n)}.$$

As expected, $T(n) \approx an + b$, where a is a per-node run-time and b is a fixed cost. Also, $T_1(n) \approx T(1)$. Thus, the percent reduction is approximately

$$\frac{T(n) - T_1(n)}{T(n)} = \frac{an + b - (a + b))}{an + b}.$$

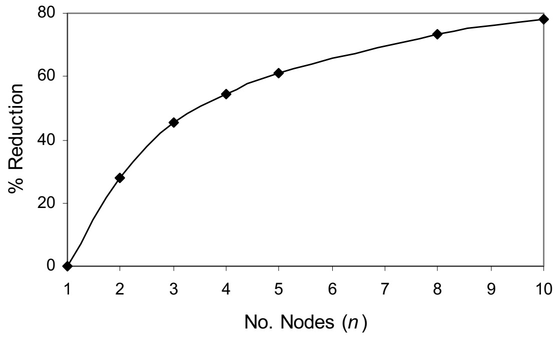

Figure 8: Reduction of Simulation Time by Clustering

For large n, this is approximately $(n-1)/n$. For example, reducing an 100-node network to a single-node network, gives a percent reduction of about 99%. In Figure 8, the percent reduction when $n = 10$ is about 80%, which is lower than $(n-1)/n = 90\%$ due to the start-up cost.

5 CONCLUSIONS

In this paper, we investigated two approximation techniques for reducing large air transportation networks to improve simulation speed: (a) Removal of low-utilization queues and (b) Removal of nodes via clustering. We tested these techniques on some simple, Jackson networks. For these examples, we found little loss in accuracy when employing the two methods. The first method appears to be quite accurate and the second method (clustering via Norton's theorem) is *exact*, provided the conditions of the theorem are satisfied.

To estimate the potential benefit in applying these ideas to a large-scale NAS model, we consider the following: The NAS consists of about 60 medium-to-large hubs, about 600 more airports with scheduled service, and about 5,000 other public-use airports. Suppose we model the low-utilization airports as single nodes (yielding, say, 5,600 nodes). The 60 medium and large hubs have more complex operations and thus require more complicated sub-models. Suppose these sub-models are 10 nodes each (for a total of 600 nodes).

Suppose the hubs carry about 75% of the operations (arrivals + departures) of the entire NAS. Let N be the total number of arrivals and departures simulated across the whole network. Since each operation to a low-utilization airport goes through a single node, and each operation to a hub goes through 10 nodes, an estimate for the simulation time of the whole network is $0.25N + (0.75N) \cdot 10 = 7.75N$ If we apply the low-utilization reduction (Section 3), the second term reduces by about 40% to $(0.25N) \cdot 0.6$. If we apply the Norton reduction technique (Section 4) to the hub airports, we reduce the simulation time of the hubs by about 80% to $(0.75N) \cdot 10 \cdot 0.2$. Thus, the entire simulation time goes from $7.75N$ to $1.65N$, for a reduction of 79% (equivalently, a speedup by a factor of 5).

Of course, in this paper, we have made *many* simplifying assumptions. The real air transportation network is not a Jackson network, so many more issues need to be addressed. For example, transitions from one node to another are governed by schedules, not random probabilities. Further, these schedules may dynamically change due to weather, crew delays, airplane maintenance, and other factors. Also, service times are not exponential. Thus, Norton-type reductions under these conditions are not guaranteed to be exact. However, we conjecture that such reductions may still be reasonably accurate under such conditions. We are currently investigating these extensions. We are also looking at ways of applying these ideas to sub-models which track weather, simulate ground delay programs, cancel flights, and resolve airplane en-route conflicts.

ACKNOWLEDGMENTS

This work has been supported in part by NASA Ames Research Center under Grant NAG-2-1565 and by FAA under Grant 00-G-016.

REFERENCES

Boucherie, R. J., and N. M. van Dijk. 1993. A generalization of Norton's theorem for queueing networks. *Queueing Systems* 13: 251-289.

Chandy, K. M., and N. D. Georganas. 1975. Decomposition and aggregation by class in closed queueing networks. *IEEE Trans. Software Eng.* 19: 36-42.

Cowie, J, H. Liu, J. Liu, D. Nicol, and A. Ogielski. 1999. Towards Realistic Million-Node Internet Simulations. In *1999 International Conference on Parallel and Distributed Processing Techniques and Applications*.

Gaier, E. M., and P. F. Kostiuk. 1998. Evaluating the economic impacts of ATM innovations on commercial air carrier operations. In *Proceedings of the 2nd USA/Europe Air Traffic Management R&D Seminar*.

Gross, D., and C. M. Harris. 1998. *Fundamentals of Queueing Theory*. 3d ed. New York, John Wiley & Sons.

Liu, B., Y. Guo, J. Kurose, D. Towsley, and W. Gong. 1999. Fluid Simulation of Large Scale Networks: Issues and Tradeoffs. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*.

Liu, Y., F. L. Presti, V. Misra, D. Towsley, and Y. Gu. 2003. Fluid Models and Solutions for Large-Scale IP Networks. To appear in *Proceedings of ACM Sigmetrics 2003*.

MacReynolds, W., and J. Sinnot. 1998. The impact of air traffic management on airspace user economic performance. In *Proceedings of the 2nd USA/Europe Air Traffic Management R&D Seminar*.

Rao, D. M., and P. A. Wilsey. 1999. Simulation of Ultra-large Communication Networks. In *International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*.

Riley, G., and M. H. Ammar. 2002. Simulating Large Networks – How Big is Big Enough? In *Proceedings of First International Conference on Grand Challenges for Modeling and Simulation*.

Yousefi, A., G. L. Donohue, and K. M. Qureshi. 2003. Investigation of En-route Metrics for Model Validation and Airspace Design Using the Total Airport and Airspace Modeler (TAAM). In *5th EUROCONTROL/FAA ATM R&D Conference*, Budapest, Hungary.

AUTHOR BIOGRAPHIES

JOHN F. SHORTLE is currently an assistant professor of systems engineering at George Mason University. His research interests include simulation and queueing applications in telecommunications and air transportation. Previously, he worked at US WEST Advanced Technologies. He recently won the Daniel H. Wagner Prize for excellence in Operations Research Practice for design of networks to offload data traffic from the PSTN. He received a B.S. in mathematics from Harvey Mudd College in 1992 and a Ph.D. and M.S. in operations research at UC Berkeley in 1996. He is a member of INFORMS and IEEE. His e-mail address is `<jshortle@gmu.edu>`.

DONALD GROSS is a Research Professor in the Department of Systems Engineering and Operations Research at George Mason University and Professor Emeritus of Operations Research, George Washington University. He is the co-author of the well-known book, Fundamentals of Queueing Theory, has numerous publications in the field of queueing theory, and is past president of INFORMS. He was Director, Operations Research and Production Systems at the National Science Foundation, 1988-1990; 1996. He has received the INFORMS Kimball Medal for Service to the Operations Research Profession. His email address is `<dgross1@gmu.edu>`.

BRIAN L. MARK is an assistant professor of electrical and computer engineering at George Mason University. His main recent research interests lie in the area of modeling and analysis of computer systems and communication network architectures. Previously he was a research staff member at NEC C&C Research Laboratories. He is a recipient of an NSF CAREER Award. He received a B.A.Sc. in computer engineering with a mathematics option from the University of Waterloo in 1991 and a Ph.D. in electrical engineering from Princeton University in 1995. He is a member of IEEE. His email address is `<bmark@gmu.edu>`.