Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

A SIMULATION BASED SCHEDULING MODEL FOR CALL CENTERS WITH UNCERTAIN ARRIVAL RATES

Thomas R. Robbins

Dept. of Marketing and Supply Chain
East Carolina University
Greenville, NC 27858, U.S.A

Terry P. Harrison

Dept. of Supply Chain and Information Systems
Penn State University
University Park, PA 16802, U.S.A.

ABSTRACT

In this paper we develop a two stage algorithm for scheduling call centers with strict SLAs and arrival rate uncertainty. The first cut schedule can be developed in less than a minute using a constructive heuristic. The schedule is then refined via a simulation based optimization approach. We find that when allowed to run for five minutes or less this two stage process can create a schedule with a total expected cost within a few percentage points of schedules generated using much more computationally intensive methods. This rapid scheduling process is designed to support front line managers who wish to evaluate multiple scheduling options in a what if analysis mode.

1 INTRODUCTION

Call centers are a large and growing component of the U.S. and world economy (Gans, Koole et al. 2003; Aksin, Armony et al. 2007) The Incoming Call Management Institute estimates that by 2008 the US will have over 47,000 call centers and employ over 2.7 million agents. Large scale call centers are technically and managerially sophisticated operations and have been the subject of substantial academic research. A call center is a facility designed to support the delivery of some interactive service via telephone communications, typically an office space with multiple workstations manned by *agents* who place and receive calls. Call center applications include telemarketing, customer service, help desk support, and emergency dispatch.

Inbound call center operations are challenging to manage; there is considerable uncertainty in estimates of arrival rates, and the operation is often subject to strict service level constraints. Our research is motivated in part by recent work with a medium sized provider of call center based technical support. While the scope of services varies from account to account, many projects are 24 x 7 support and virtually all are subject to some form of Service Level Agreement (SLA). There are multiple types of SLAs, but the most common specifies a minimum level of the Telephone Service Factor (TSF). A TSF SLA specifies the proportion of calls that must be answered within a specified time. For example, an 80/120 SLA specifies that 80% of calls must be answered within 120 seconds. A very important point is that the service level applies to an *extended period*, typically a week or a month, which appears to be common practice in this segment of the industry. The SLA does not define requirements for a day or an hour; so the desk is typically staffed so that at some times the service level is underachieved, sometimes overachieved, and is on target for the entire month. The outsourcing contract often specifies substantial financial penalties for failing to meet the SLA.

The key difficulty involved with staffing this call center is a fixed SLA with a variable and uncertain arrival rate pattern. The number of calls presented in any ½ hour period is highly variable with multiple sources of uncertainty. In addition to day of week seasonality these call center projects also experience very significant time of day seasonality.

The staffing challenge in this call center is to find a minimal cost staffing plan that achieves the global SLA target with a high probability. The schedule must obviously be locked in before arrival rate uncertainty is revealed. While management has some recourse to adjust manpower during the course of the day (overtime, early dismissal) these actions are generally very limited.

Standard call center scheduling models typically enforce the service level in every time period and therefore often over staff the call center. Most models also fail to adequately address the uncertainty in inbound call volume and tend to under staff in volatile periods. Managers often manually adjust the schedules based on experience to compensate for these shortcomings, making extensive optimization moot. Several recent scheduling models attempt to address this uncertainty, but result in computationally intensive algorithms. We develop an algorithm that can develop near optimal schedules very quickly to support interactive planning.

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

2 LITERATURE REVIEW

Detailed summaries of the call center oriented literature are provided in (Gans, Koole et al. 2003) and (Aksin, Armony et al. 2007). Many call center scheduling models are derivatives of the set covering model developed in (Dantzig 1954). Dantzig's model assumed by period staffing requirements were defined exogenously, as were a set of feasible schedules. His algorithm seeks the lowest cost covering of the requirements. The set covering model is often formulated as a Mixed Integer Program and becomes intractable as the number of schedules becomes large, which is common if operations are continuous and breaks are scheduled explicitly. Many models implement a two phased approach where breaks are ignored in the first phase and are then scheduled in a second pass. This general approach is outlined in (Pinedo 1995) and examined in more detail in (Thompson 1995; Aykin 1996; Brusco and Jacobs 2000).

More recently attention has been focused on the issue of uncertainty in arrival rates. Statistical analysis of call center data in (Brown, Gans et al. 2005; Robbins 2007) supports the notion that arrival rates are uncertain. (Bassamboo, Harrison et al. 2005) develop a model that attempts to minimize the cost of staffing plus an imputed cost for customer abandonment for a call center with multiple customer and server types when arrival rates are variable and uncertain. (Harrison and Zeevi 2005) solve the staffing problem for call centers with multiple call types, multiple agent types, and uncertain arrivals using a fluid approximation.

(Atlason, Epelman et al. 2007) develop an algorithm that combines server sizing and staff scheduling into a single optimization problem. This model focuses on the impact that staffing in one time period can affect performance in the subsequent period, a fact ignored in many models. The algorithm utilizes discrete event simulation to calculate service levels under candidate staffing models, and a discrete cutting plane algorithm to search for improving solutions. This approach has the advantage of making accurate service level calculations for call centers with non-stationary arrival patterns, but this comes at a high computational cost. (Robbins and Harrison 2008) develop a modified version of the set covering approach that works to a globally optimum schedule when arrival rates are uncertain. This model is formulated a stochastic mixed integer program and requires significant computational effort to solve.

3 SCHEDULING MODEL

The objective of this paper is to develop a scheduling model that can operate in an interactive decision support framework. As such it should allow for very rapid development of good schedules to support interactive planning in a *what-if* approach. It should then allow schedules to be refined and optimized. Finally it should operate on a standard desktop computer to allow front line supervisors to run it interactively.

To accomplish this objective we developed a two stage process. In the first stage we use a constructive heuristic to incrementally develop a preliminary schedule that will satisfy the aggregate service level requirements based on a stationary period by period approximation of the service level. In the second stage we evaluate the schedule via discrete event simulation and use a local search algorithm to find lower cost schedules.

3.1 Constructive Heuristic

The constructive heuristic builds a schedule using a greedy algorithm; it iteratively adds assignments that cover the highest expected number of calls answered outside of the service level requirement. To define the algorithm more formally we use the following notation.

Sets

I: time periods
J: possible schedules
K: scenarios

Decision Variables

x_j: number of resources assigned to schedule j

Deterministic Parameters

a_{ij}: a 1 if schedule j is staffed in time i, 0 otherwise
g: global SLA goal
d: allowable delay before a call must be answered
μ: minimum number of agents in any time period
r: per point penalty cost of TSF shortfall

State Variables

S_i: Average TSF shortfall in period i
b_j: shortfall calls covered by schedule j
C_j: Cover for schedule j
A: the expected aggregate service level
O: the service level offset

Stochastic Parameters

n_{ik}: number of calls in period i of scenario k

Each scenario in the set K represents a vector of call volumes per time period in the planning horizon. These call volumes can be generated using any approach that can characterize the variability of the call stream. In our approach we assume that the call volume on any given day is a normally distributed random variable. We further assume that the proportion of calls received during any 30 minute period is a normally distributed random variable. We assume the parameters for each 30 minute period are consistent across all days of the weeks. The call generation algorithm is detailed in (Robbins 2007).

The average shortfall in each time period is calculated by:

$$S_i = \frac{\sum_{k \in K} \left[n_{ik} \left(1 - TSF\left(n_{ik}, \sum_{j \in J} a_{ij} x_j, d \right) \right) \right]}{K} \quad (1)$$

The function $TSF(v, n, d)$ calculates the telephone service factor, the proportion of calls answered within *d* seconds, if call volume is *v*, and the number of agents is *n*. This calculation is based on the Erlang A model and implemented using the fluid approximations defined in (Garnett, Mandelbaum et al. 2002). The measure S_i is the average number of calls that are not answered within the allowable delay for time period *i*. Periods with a large number of out of service level calls are good candidates for incremental staffing.

To determine where to best add incremental staffing we calculate the *cover* for each feasible schedule. The cover is defined as the average number of out of service level calls presented to this schedule per period.

$$C_j = \frac{\sum_{i \in I} S_i a_{ij}}{\sum_{i \in I} a_{ij}} \quad (2)$$

The next schedule to staff is selected as the schedule with the maximum cover.

Based on these definitions we can outline a basic initial staffing heuristic as follows.

1. Initialization
   a. Generate a set of *K* call volume scenarios
   b. Create an initial schedule with no staffing
   c. Define a target service level scheduling bias
2. Repeat the following until `Stop=True`
   a. Find the schedule in *J* with maximum cover C_j and increment staffing for that schedule.
   b. Calculate the per period out of service level call volume S_i
   c. Calculate *A*, the aggregate expected service level, *i.e.* the average proportion of calls answered with *d* seconds.
   d. If $A \geq g + O$ then `Stop = True`

Figure 3-1 Staffing Heuristic

The algorithm continues until the estimated aggregate service level reaches the goal level plus an offset. The offset parameter is a factor that allows the stopping point to be biased relative to the goal. Our empirical results show that better schedule can be found by applying a small offset that varies based on the seasonality of the project. The offset allows for an adjustment to compensate for any bias in the stationary period by period based service level approximation.

3.1.1 The Minimum Staffing Problem

The algorithm outlined in Figure 3-1 can quickly generate a schedule that will satisfy the aggregate service level requirement, however the resulting schedule may not meet minimum staffing requirements. A minimum staffing requirement necessitates a minimum number of agents that must be scheduled at all times. In the operation we analyzed a minimum of two agents are required at all times.

Because operations are 24 × 7 and overnight call volume tends to be quite low on some projects, the staffing heuristic will often leave portions of the overnight period with no staffing, a solution that is clearly unacceptable.

To address the minimum staffing issue we modified the cover calculation to add a bonus for covering time periods with staffing levels below the minimum. In practice, setting this bonus parameter turned out to be very important in terms of the quality of the final schedule generated. If the parameter was set too high, off peak hours are scheduled too early in the process and the resulting schedule is too costly. If set too low then the schedule will reach the aggregate service level goal without satisfying the minimum staffing constraint.

After extensive experimentation we settled on a modified algorithm defined as follows.

1. Initialization
   a. Set the min staffing cover coefficient to an initial value
   b. Set `Stop = False`
2. Search: repeat until `Stop=True`
   a. Execute the staffing algorithm outline in Figure 3-1.
   b. If all min staffing constraints are satisfied the `Stop = True`, else increment the min staffing cover coefficient

Figure 3-2 Modified Staffing Heuristic

3.2 Simulation Based Search

In this stage we use Discreet Event Simulation (DES) to model the operation of the call center. Unlike the analytically based model used so far, the DES approach simulates individual call processing. The simulation model is designed to generate call arrivals using the same statistical model used to generate call scenarios for the initial heuristic. The simulation model also varies the number of agents based on the time phased staffing model generated from the scheduling model.

Basic DES can evaluate the expected outcome of the candidate schedule, but in order to find a better schedule, we need to implement some form of optimization algorithm. We implement a local search algorithm that starts with the schedule generated from the heuristic and searches the neighborhood of closely related schedules. The local search algorithm is guided by a variable neighborhood

search (VNS) metaheuristic. VNS is a metaheuristic that makes systematic changes in the neighborhood being searched as the search progresses (Hansen and Mladenovic 2001; Hansen and Mladenovic 2005). The general structure of the VNS is as follows:

1. Initialization
   a. Select the set of neighborhood structures N_k, for $k = 1,...,k_{\max}$
   b. Construct an initial incumbent solution, x_I, using a heuristic procedure.
   c. Select a confidence level α for the selection of a new incumbent solution
2. Search: repeat the following until `Stop=True`
   a. Set $k = 1$
   b. Find $n_{k_{\min}}$ candidate solutions, x_C that are neighbors of x_I
   c. Simulate the system with each candidate and compare the results to the incumbent using a pair wise t Test at the α level of significance.
   d. If any x_C is superior to x_I at the α level then set $x_I = x_C^*$, where x_C^* is the best candidate solution
   e. Else, set $i = n_{k_{\min}}$, set `Found = False`, and repeat until ($i = n_{k_{\max}}$ or `Found=True`)
      i. Find a new candidate x_{k_i}
      ii. Simulate the system with each candidate and compare the results using a pairwise T Test.
      iii. If x_{k_i} is superior to x_I at the α level then set $x_I = x_{k_i}$ and found = True
   f. If a no new incumbent was found in neighborhood k then
      i. set $k = k + 1$
      ii. if $k > k_{\max}$ then Stop = True

A common approach in VNS is to define a series of nested neighborhood structures such that

$$N_1(x) \subset N_2(x) \subset ... \subset N_{k_{Max}}(x) \quad \forall x \in X$$

When defining the neighborhood structure we make the distinction between the set of active schedules, those schedules with a non-zero assignment in the candidate schedule, and feasible schedules which include all schedules in the feasible schedule set. Based on this distinction we define the following neighborhoods

- $N_1(x)$: **Active 1 Change**: the set of all staff plans where an active schedule is either incremented or decremented by one.
- $N_2(x)$: **Active 2 Change**: pick any two active schedules and independently increment or decrement each.
- $N_3(x)$: **Feasible 1 Change**: pick any feasible schedule and add an assignment.
- $N_4(x)$: **Feasible 2 Change**: pick any feasible schedule and add an assignment, pick an active schedule and decrement the number assigned.

4 NUMERICAL EVALUATION

To test our algorithm we evaluate it against models developed for three real world IT support projects. We refer to these projects as Project J, Project S and Project O. Each project has unique characteristics that create different seasonality patterns and different levels of variability.

Project J is an IT help desk supporting approximately 30,000 corporate used. This help desk receives about 16,000 calls per month and exhibits a strong weekly and daily seasonality pattern. Call volume exhibits a strong spike in the morning between 8 and 11 AM and a smaller peak in the afternoon between 1 and 3 PM. Project O supports a corporate and store employees for a large retail chain. Call volume is slightly lower, at about 15,000 calls per month and exhibits a less dramatic daily seasonality pattern. Call volume peaks around 8AM and declines steadily throughout the day, without the narrow volume peak exhibited in Project J. Project S also supports retail and corporate users and has a seasonality pattern similar to Project O. Call volume is much higher, at about 45,000 calls per month and is much more volatile than project O. More detailed descriptions of these projects, and the parameters of the call volume models can be found in (Robbins 2007).

For each project we develop schedules for five variability options; these options allow for various levels of part-time staffing. The least flexible option, A, allows only for five day a week eight hour shifts. With these restrictions there are 336 unique schedules to which agents can be assigned. The most flexible option, E, allows five day a week four hour shifts and expands the total number of feasible schedules to 3,696. The scheduling options are summarized in Table 1.

Table 1: Scheduling Patterns

Pattern	Schedule Types Included	Feasible Schedules
A	5x8 only	336
B	5x8, 4x10	1,680
C	5x8, 4x10, 4x8	3,024
D	5x8, 4x10, 4x8, 5x6	3,360
E	5x8, 4x10, 4x8, 5x6, 5x4	3,696

We evaluated each of the three project against the five schedule options for a total of 15 test cases. In each case we benchmarked our solution against the solution generated using the mixed integer stochastic program analyzed in (Robbins 2007). That algorithm generates an optimal solution based on an analytical approximation of service levels using an independent period by period approximation, but the resulting solution may not be optimal given interactions between periods and errors due to the analytic approximation of the service level. Solution times for this algorithm range from 45 to 120 minutes. Table 2 summarizes the results.

Table 2: Objective Value Comparison

	Simulated Total Cost					
	Heuristic Only	5 Min Simulation Search	Full Simulation Search	Stochastic Prog. Solution	5 Min Gap	Full Search Gap
Project J						
Sched A	12,806	12,156	12,126	12,168	-0.10%	-0.34%
Sched B	12,069	11,825	11,759	12,157	-2.73%	-3.28%
Sched C	12,373	12,088	11,923	11,773	2.68%	1.27%
Sched D	12,193	11,813	11,709	11,618	1.68%	0.79%
Sched E	12,011	11,930	11,930	11,457	4.13%	4.13%
Project S						
Sched A	33,652	31,544	30,142	31,203	1.09%	-3.40%
Sched B	34,139	30,510	29,235	30,667	-0.51%	-4.67%
Sched C	33,856	30,472	29,283	30,917	-1.44%	-5.29%
Sched D	33,448	31,347	29,150	30,064	4.27%	-3.04%
Sched E	33,800	32,279	29,221	30,416	6.13%	-3.93%
Project O						
Sched A	13,004	12,539	12,216	12,141	3.28%	0.62%
Sched B	13,026	12,498	12,019	12,215	2.31%	-1.60%
Sched C	12,990	12,441	12,193	12,138	2.50%	0.45%
Sched D	12,805	12,193	12,071	12,010	1.52%	0.50%
Sched E	12,858	12,341	12,140	11,974	3.06%	1.38%

We tuned the algorithm to optimize the schedule that was generated after the staffing heuristic had run and the simulation search had run for a few minutes. Testing revealed that this was best accomplished by setting the offset used in the algorithm of Figure 3-1to 5%. A smaller bias improves the quality of the solution found in the heuristic, but makes the simulation search more difficult. We found that the simulation search finds improvements solution faster when it is offset because improving single changes are easier to find than improving two changes.

Overall we found that our algorithm can find solutions that evaluate within a few points of solution found by the stochastic integer program within the five minute target time frame, in some cases better than the solution found by the stochastic program. (Recall that since the stochastic program uses an analytical approximation to estimate the service level, it's solution is not necessarily optimal.) In the worst case the five minute solution is 6% more expensive than the stochastic programming solution; in the best case it is 2.7% less expensive. If allowed to run until it reaches its heuristic stopping rule, it often finds a better solution than the stochastic program, and in all but one case finds a solution within 1.5%. This verifies our hope that this algorithm can be used to find a good solution fast and if allowed to run for a longer period of time can find a near optimal solution. This supports the goal of interactive what-if analysis of schedule options – followed by a more detailed search for a final schedule.

Figures 1-3 show how the objective value of the incumbent solution evolves for each project for the B scheduling option. (Schedule option B in general had the most iterations.)

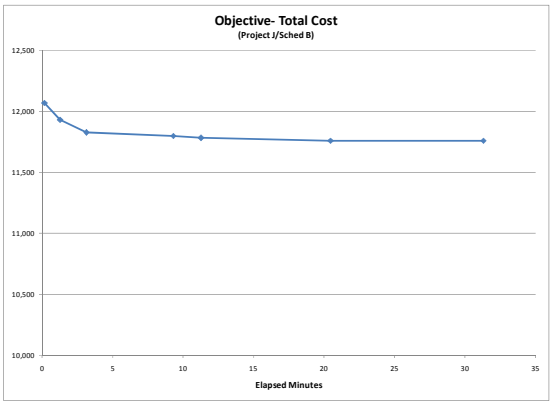

Figure 1: Objective Project J/Schedule B

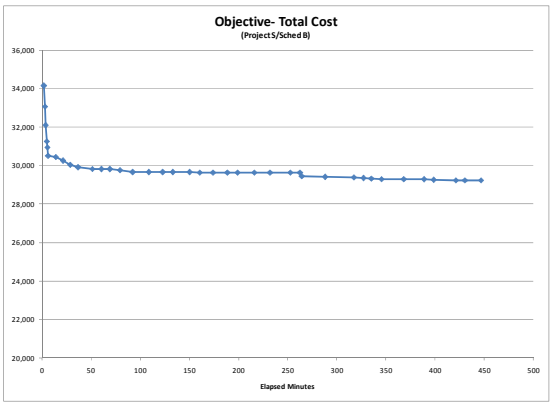

Figure 2: Objective Project S/Schedule B

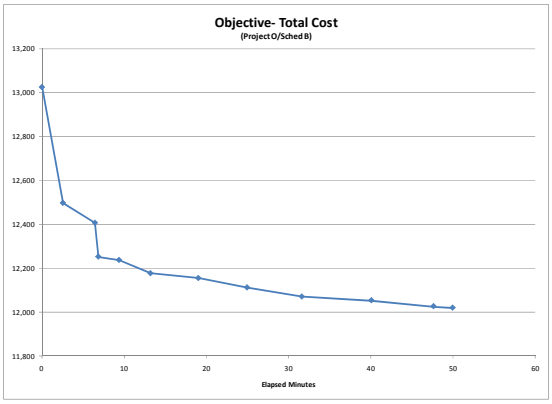

Figure 3: Objective Project O/Schedule B

In each case the heuristic finds a reasonable solution that is quickly improved by the simulation based search. The degree and frequency of the improvement then drops off rapidly. In the case of Project J and Project O the algorithm terminates relatively quickly with an objective near the objective of the stochastic integer program. In the case of Project S the algorithm runs for an extended period of time, but always finds a better solution than the stochastic integer program. We believe this is due to the fact that call volume drops off slowly in Project S so there are many schedules that are near optimal. That relatively higher peaks in Projects J and O create a smaller set of near optimal solutions.

5 SUMMARY AND CONCLUSIONS

In this paper we outline an algorithm and preliminary results for a call center scheduling model that can be run by a front line manager on a desktop computer. We seek an algorithm that allows the manager to quickly evaluate different scheduling options and find that in relatively short periods of time we can generate quality schedules.

Because our algorithm is based on simulation and a search heuristic we can make no claims about optimality, but comparing the results to a optimal analytical approximation shows favorable results.

More detailed and more structured testing of our algorithm is required to better understand the tradeoffs made when setting key tuning parameters such as the initial heuristic offset.

6 REFERENCES

Aksin, Z., M. Armony and V. Mehrotra 2007. The Modern Call Center: A Multi-Disciplinary Perspective on Operations Management Research. *Production and Operations Management* **16**(6) p. 665-688.

Atlason, J., M. A. Epelman and S. G. Henderson 2007. Optimizing Call Center Staffing Using Simulation and Analytic Center Cutting-Plane Methods. *Management Science* p. 15.

Aykin, T. 1996. Optimal Shift Scheduling with Multiple Break Windows. *Management Science* **42**(4) p. 591-602.

Bassamboo, A., J. M. Harrison and A. Zeevi 2005. Design and Control of a Large Call Center: Asymptotic Analysis of an LP-based Method. Working Paper 51p.

Brown, L., N. Gans, A. Mandelbaum, A. Sakov, S. Haipeng, S. Zeltyn and L. Zhao 2005. Statistical Analysis of a Telephone Call Center: A Queueing-Science Perspective. *Journal of the American Statistical Association* **100**(469) p. 36-50.

Brusco, M. J. and L. W. Jacobs 2000. Optimal Models for Meal-Break and Start-Time Flexibility in Continuous Tour Scheduling. *Management Science* **46**(12) p. 1630-1641.

Dantzig, G. B. 1954. A Comment on Edie's "Traffic Delays at Toll Booths". *Journal of the Operations Research Society of America* **2**(3) p. 339-341.

Gans, N., G. Koole and A. Mandelbaum 2003. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management* **5**(2) p. 79-141.

Garnett, O., A. Mandelbaum and M. I. Reiman 2002. Designing a Call Center with impatient customers. *Manufacturing & Service Operations Management* **4**(3) p. 208-227.

Hansen, P. and N. Mladenovic 2001. Variable neighborhood search: Principles and applications. *European Journal of Operational Research* **130**(3) p. 449-467.

Hansen, P. and N. Mladenovic (2005). Variable Neighborhood Search. Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. E. K. Burke and G. Kendall. New York, NY, Springer**:** 211-238.

Harrison, J. M. and A. Zeevi 2005. A Method for Staffing Large Call Centers Based on Stochastic Fluid Models. *Manufacturing & Service Operations Management* **7**(1) p. 20-36.

Pinedo, M. 1995. *Scheduling: theory, algorithms, and systems*, Prentice Hall. Englewood Cliffs, N.J.

Robbins, T. R. 2007. Managing Service Capacity Under Uncertainty - Unpublished PhD Dissertation (http://www.personal.psu.edu/faculty/t/r/trr147). Working Paper 241p.

Robbins, T. R. and T. P. Harrison 2008. A Stochastic Programming Model for Scheduling Call Centers with Global Service Level Agreements. Working Paper 34p.

Thompson, G. M. 1995. Improved Implicit Optimal Modeling of the Labor Shift Scheduling Problem. *Management Science* **41**(4) p. 595-607.

AUTHOR BIOGRAPHIES

THOMAS ROBBINS recently earned a PhD in Business Administration and Operations Research from Penn State University. Prior to beginning his academic career he worked in professional services for approximately 18 years. He holds an MBA from Case Western Reserve and a BSEE from Penn State. He currently teaches statistics and operations management at Penn State. He is currently an Assistant Professor at East Carolina University. His email address is <robbinst@ecu.edu>.

TERRY P. HARRISON is the Earl P. Strong Executive Education Professor of Business and Professor of Supply

Chain and Information Systems at Penn State University. He holds a Ph.D. and M.S. degree in Management Science from the University of Tennessee and a B.S in Forest Science from Penn State. He was formally the Editor-in-Chief of *Interfaces* and is currently Vice President of Publications for INFORMS. His mail address is <tharrison@psu.edu>.