Paper SAS5480-2016

Using SAS® Simulation Studio to Test and Validate SAS/OR® Optimization Models

Ed Hughes, Emily Lada, Leo Lopes, and Imre Pólik, SAS Institute Inc.

ABSTRACT

In many discrete-event simulation projects, the chief goal is to investigate the performance of a system. You can use output data to better understand the operation of the real or planned system and to conduct various "what-if" analyses. But you can also use simulation for validation—specifically, to validate a solution found by an optimization model.

In optimization modeling, you almost always need to make some simplifying assumptions about the details of the system you are modeling. These assumptions are especially important when the system includes random variation—for example, in the arrivals of individuals, their distinguishing characteristics, or the time needed to complete certain tasks. A common approach holds each random element at some nominal value (such as the mean of its observed values) and proceeds with the optimization.

You can do better. In order to test an optimization model and its underlying assumptions, you can build a simulation model of the system that uses the optimal solution as an input and simulates the system's detailed behavior. The simulation model helps determine how well the optimal solution holds up when randomness and perhaps other logical complexities (which the optimization model might have ignored, summarized, or modeled only approximately) are accounted for. Simulation might confirm the optimization results or highlight areas of concern in the optimization model.

This paper describes cases in which you can use simulation and optimization together in this manner and discusses the positive implications of this complementary analytic approach. For the reader, prior experience with optimization and simulation is helpful but not required.

INTRODUCTION

SAS® Simulation Studio, a component of SAS/OR® software, provides an interactive graphical environment for building, running, and analyzing discrete-event simulation models. In its own right, discrete-event simulation provides extremely valuable insights into the operation and performance of complex systems across a range of scenarios. Using it to complement optimization modeling adds even more value. Discrete-event simulation can evaluate the simplifying assumptions about a system's detailed operations that are necessary for optimization modeling, either confirming the optimization model's validity or identifying elements of the model that need correction.

This paper begins with a look at both optimization modeling and discrete-event simulation modeling, and explores how they can most effectively work together to create additional analytic value. It then considers two examples of a combined optimization and simulation approach and discusses the resulting benefits.

CHOOSING HOW TO MODEL COMPLEX SYSTEMS

Complex, workflow-oriented systems are found in many practical settings. Examples include customer service in retail, patient treatment in hospital emergency departments and intensive care units, parcel movement by shipping companies, and claims processing by government agencies and insurance companies. In every instance, the appropriate allocation and use of available resources are important to efficient operation of the system, as is intelligent configuration of the system's workflow. Optimization modeling and discrete-event simulation modeling each have important roles to play. Together they can contribute even more.

OPTIMIZATION MODELING

Optimization models consist of three major components. Each decision that must be made is represented by a *decision variable* that might assume any value in a range (continuous), take only integer values (integer), or be limited to the values 0 and 1 (binary). You use functions of the decision variables to

construct the other two model elements: the *constraints*, which describe restrictions on the decisions, and the *objectives*, which describe goals to be achieved. Constraints include budgets and resource limits, regulations, and other restrictions. Common objectives include maximizing profit or revenue and minimizing distance, cost, or waste. Optimization model types are defined by the types of decision variables that are used and the types of functions that are used to define constraints and objectives. Good optimization models are data-driven so that it's easy to update critical information and reuse models to solve a series of similarly structured problems.

Optimization proactively recommends solutions (sets of decisions) for consideration, and you can implement these recommendations directly or view them simply as additional alternatives to consider. Because it takes an active role in the decision process, optimization is referred to as an instance of "prescriptive analytics." In addition, the objective value of an optimal solution provides a benchmark for solutions from other sources.

Optimization models, like all analytic models, require certain compromises as you translate from the real world to the model. Not all details are included, for a number of reasons. You might not have data available for some elements, and the precision of the available data also limits the amount of detail you should include in the model. A highly detailed model might require too much time for optimization. An excessively detailed solution might be impractical to implement.

Additionally, optimization modeling by its very nature does not and cannot account for minute details of the operation of a workflow system. Optimization is designed to consider and improve system performance as measured over a period of time, and accordingly it aggregates performance data over that time period; it doesn't measure or track every individual action or event. It cannot account for best-case or worst-case scenarios. It cannot account fully for interactions between individuals present in a system. Stochastic optimization can account for some of the effects of random variation present in a system, but it cannot fully capture the effects of interactions between multiple sources of variation.

To capture and gather data on the finer details of system performance, you need a different modeling approach, such as discrete-event simulation.

DISCRETE-EVENT SIMULATION MODELING

In addition to fostering better understanding of the system being modeled (a benefit of all analytic modeling), discrete-event simulation primarily enables you to generate detailed data on the performance of a system in selected scenarios. As the name suggests, this type of simulation is driven by the occurrence of events such as arrivals to or departures from the system, the start or conclusion of some service, or system resources changing their operational status. The data that a discrete-event simulation model generates describe each event that occurs in a system during the period being simulated, because the model is expressly designed to simulate each event and its effects. Discrete-event simulation also models interactions and random variation directly and in detail. This modeling technique is applied in many situations involving individuals or items that move through a workflow process and receive some sort of service, assistance, or modification at one or more points in the process by interacting with the system's resources. Manufacturing systems, customer service operations, medical treatment, and many other similar workflow-oriented systems can be modeled in great detail with discrete-event simulation.

Discrete-event simulation depicts each step of an individual's or object's journey through a workflow structure. The individual or object, which is referred to as an entity, can interact not only with system resources that provide service or assistance but also with other entities. On the simplest level, entities affect each other's path through the system just by their presence. For example, an entity that enters a queue where many other entities are waiting is much more likely to wait longer and to leave the queue prematurely than an entity that arrives at an empty queue. Just as with other aspects of system operation, discrete-event simulation can capture many details of entity interactions as it models them. The duration, the nature, or even the existence of interactions with other entities and with system resources can be influenced by an entity's attributes, by random variation, by the current status of the system, and even by the time of day or day of the week.

Discrete-event simulation's primary role is to answer what-if questions about system performance in various scenarios. Consequently, it is largely a predictive form of analytics, in contrast to the

predominantly prescriptive role played by optimization. Simulation does not directly recommend any set of actions, but predicts what would happen if a specified set of actions were chosen. Simulated data can supply input to response surface approaches to determining optimal decisions, in which a response surface is fitted to the simulated data and is then used to determine an optimal set of decisions.

Discrete-event simulation raises several of the same data and modeling issues as optimization, but its primary focus on the details of system performance makes it better equipped than optimization to model detailed system metrics.

OPTIMIZATION PLUS DISCRETE-EVENT SIMULATION

Optimization and discrete-event simulation each make distinct and significant contributions to the analysis and improvement of the operations of complex systems. As noted previously, their contributions are largely complementary: optimization supplies prescriptive analytics based on aggregated performance, and discrete-event simulation supplies detailed predictive analytics.

Together, optimization and discrete-event simulation can contribute even more to improving system performance than they can separately. A blog post (Lada 2015a) discusses the complementary benefits of these two approaches. On the simplest level, optimization provides a set of decisions regarding system configuration, resource allocations, and other choices of interest, and discrete-event simulation confirms the optimality of these decisions in the form of detailed performance statistics for multiple possible scenarios.

Recall that optimization modeling requires certain simplifying assumptions to aggregate system performance and that other assumptions are sometimes needed for tractability. You can use discrete-event simulation to test many of these assumptions. Although the primary role of optimization is prescriptive, it also plays an implicit predictive role via its objective function: if you take the actions prescribed by an optimal solution, the optimization model predicts that the outcome will be the optimal value of the objective function. Because this predictive element of optimization corresponds directly to the primarily predictive mission of discrete-event simulation, it creates an opportunity to use simulation to test and validate (or invalidate) optimization results by comparing simulated results to the optimal objective function value. Examples of objective functions include total revenue, costs, and production levels.

Discrete-event simulation can also test whether constraints that you include in the optimization model are violated, either in an aggregated view of events or on a transient basis, at one or more points during the simulation run. For example, a constraint in an optimization model might limit resource use, averaged over a period of time. A discrete-event simulation model run might show violation or satisfaction of that average limit, or it might show compliance with the average limit but an uncomfortably large number of instances where instantaneous resource use exceeds the limit during the run.

If you use an optimal solution as an input to a discrete-event simulation model of the same system, and the simulated data and metrics show any statistically significant deviation from the optimal objective function value, then you need to adjust or correct one or both of the models. You can begin by focusing on the corresponding areas of the two models that produce the most significantly divergent results. Because discrete-event simulation modeling is by design more detailed than optimization modeling, it is reasonable (assuming that both models have passed basic validity tests) to focus primarily on questioning, adjusting, and improving the optimization model (but don't rule out questioning both models). Conversely, simulated results that show no statistically significant deviation from your optimization results should increase your confidence in the validity of both the optimal solution and the optimization model that produced it.

Moreover, adding discrete-event simulation to optimization enables you to account in detail for the queueing aspects of the system that you are modeling. Optimization modeling, with its longer view of system performance, does not account for queueing, because doing so requires continuous monitoring, not aggregate metrics. Assuming that a given number of entities arrive over a period of time, for example, is qualitatively different from executing each of those arrivals according to a realistic arrival process.

Execution of a discrete-event simulation model creates and plays out the simulated events that update the status of both the queues in the system and the entities that populate the queues, and provides data you can use to compile queueing statistics. Discrete-event simulation can measure time-persistent

statistics like resource utilization levels, queue lengths, and the number of entities in the system, and observation-based statistics such as the time an entity spends in a queue or a section of the system. By playing out each modeled scenario event by event, discrete-event simulation can record metrics from best-case scenarios, worst-case scenarios, and other scenarios of interest.

One presumed alternative to discrete-event simulation modeling is the direct application of queueing theory, which has produced extensive results describing queue length, waiting time, and similar metrics. Unfortunately, queueing theory is usually laden with simplifying assumptions that most real-world systems violate. Additionally, queueing theory focuses primarily on long-term, steady-state behavior of queueing systems. Many real-world queueing metrics of interest concern periods of system operation that are too brief to permit systems to reach steady state. A recent blog post (Lada 2015b) compares queueing theory and discrete-event simulation results and finds the latter to be much more useful in a practical setting.

EXAMPLES

Two sample projects illustrate the advantages of combining optimization with PROC OPTMODEL in SAS/OR and discrete-event simulation with SAS Simulation Studio in modeling a complex system. The first example is hypothetical and is based on careful observation of food truck operations by two of the authors. The second example is factual and recounts key elements of a project undertaken by SAS in partnership with a European utility company.

EXAMPLE 1: FOOD TRUCK PRICING AND STAFFING

In this example, a food truck operator uses both optimization and discrete-event simulation to plan pricing and staffing for what is anticipated to be a period of heavy demand. An optimization model generates optimal pricing and staffing decisions that are input to a discrete-event simulation model of the food truck and its customers. Discrete-event simulation incorporates and highlights the queueing elements of the decision problem that the optimization approach does not model directly. Analysis of the simulated data evaluates the optimization model's assumptions and conclusions, and broadens the scope of the evaluation.

Background

Food trucks are a popular manifestation of the long-established tradition of food vendors working on urban streets. In addition to individual food trucks serving tourists, festival-goers, and office workers, "food truck rodeos," which are well-organized gatherings of food trucks, are growing in popularity. Food trucks as a whole, and especially those at food truck rodeos, have moved away from selling typical street-vendor fare and now focus increasingly on providing specialized and gourmet entrées, snacks, and desserts.

One food truck is planning for a traditionally well-attended food truck rodeo that will run from 11 a.m. to 5 p.m. Although this food truck is fairly new to the market, indications are that its single product, a plate of two freshly prepared vegetarian spring rolls, should be in relatively high demand, with an average of 30 customers per hour expected. The truck's proprietor wants to ensure that she prices the product appropriately and plans staffing carefully in order to maximize profits and further her truck's growing reputation. She is considering pricing options for her spring roll plate, ranging from $2.00 to $2.75 in 25-cent increments. Depending on the price of the plate, each customer might order up to four plates at a time. The truck can accommodate up to three workers in addition to the proprietor, who is always present and working when the truck is operating. Workers are paid $8 per hour. The proprietor does not pay herself an hourly wage.

Optimization Model

This problem can be modeled as a mixed integer linear program (MILP) with PROC OPTMODEL in SAS/OR. A SAS program that builds and solves this optimization model appears in the appendix. An integer decision variable represents the possible staffing levels (0, 1, 2, and 3), and a set of binary decision variables (one for each candidate price) represents the choice of price. A constraint requires the binary price decision variables to sum to 1, meaning that exactly one price must be selected. Another

constraint requires that the staff on hand (plus the proprietor) be able to produce enough plates to meet expected hourly demand. The objective maximizes profit, measured as revenue minus staffing costs.

In building the optimization model, you aggregate and make a number of simplifying assumptions. You can assume that the average number of customers (30) arrives each hour, corresponding to 180 customers in six hours. You can also assume that the order sizes obey the empirical distribution shown in Table 1, but because the optimization model views demand only in the aggregate, it uses only the expected order size figures shown in the bottom row of Table 1.

Price / Quantity	Probability of Ordering the Indicated Number of Plates			
	$2.00	**$2.25**	**$2.50**	**$2.75**
1	0.1	0.1	0.3	0.4
2	0.2	0.3	0.4	0.3
3	0.3	0.4	0.2	0.2
4	0.4	0.2	0.1	0.1
Exp. Order Size	3.0	2.7	2.1	2.0

Table 1. Empirical Distributions of Order Size

To calculate the number of plates ordered each hour, multiply the expected order size by 30, the expected number of customers per hour. For example, at a price of $2.00 the expected order size is 3 plates. The optimization model assumes that each of the 30 customers who arrive each hour will order 3 plates, indicating demand for 90 plates per hour (a total of 540 plates over the six-hour period). The proprietor and her staff must be able to produce at least 90 plates per hour. The proprietor, who is more experienced, can produce 30 plates per hour, and each of her staff members can produce 20 plates per hour.

The optimization model recommends a price of $2.25 and three staff (plus the proprietor), yielding a profit of $949.50 in six hours. For comparison, you can solve this model additional times, fixing the selection of each candidate price in turn. The results are shown in Table 2.

Price	Optimal Staffing	Profit
$2.00	3	$936.00
$2.25	3	$949.50
$2.50	2	$849.00
$2.75	2	$894.00

Table 2. Optimal Staffing and Profit for Each Candidate Price

Even with this broader approach, the optimization model glosses over the queueing aspects of this problem by aggregating performance over the six-hour period. Traffic at food truck rodeos can vary greatly in intensity. Therefore, assuming that customers will arrive at the same steady average rate every hour is very likely to be unrealistic. It would also be enormously helpful to know more about how long the line at the truck is likely to be during the rodeo. Customers do not have infinite patience, especially with other food trucks nearby, and an excessively long line could easily result in lost business, reduced profits, and loss of goodwill.

Discrete-Event Simulation Model

Because queueing considerations play such a large role in food truck performance, discrete-event simulation is an appropriate modeling approach. The discrete-event simulation model of the food truck, which is built using SAS Simulation Studio in SAS/OR, is fairly simple but captures many of the details that the optimization model overlooks. The compound blocks highlight the major phases of the model:

customers arriving, being served, and departing. Total revenue is updated as each customer departs. At the end of the food truck rodeo, total profit is calculated. Figure 1 shows this model in a Model window in SAS Simulation Studio.

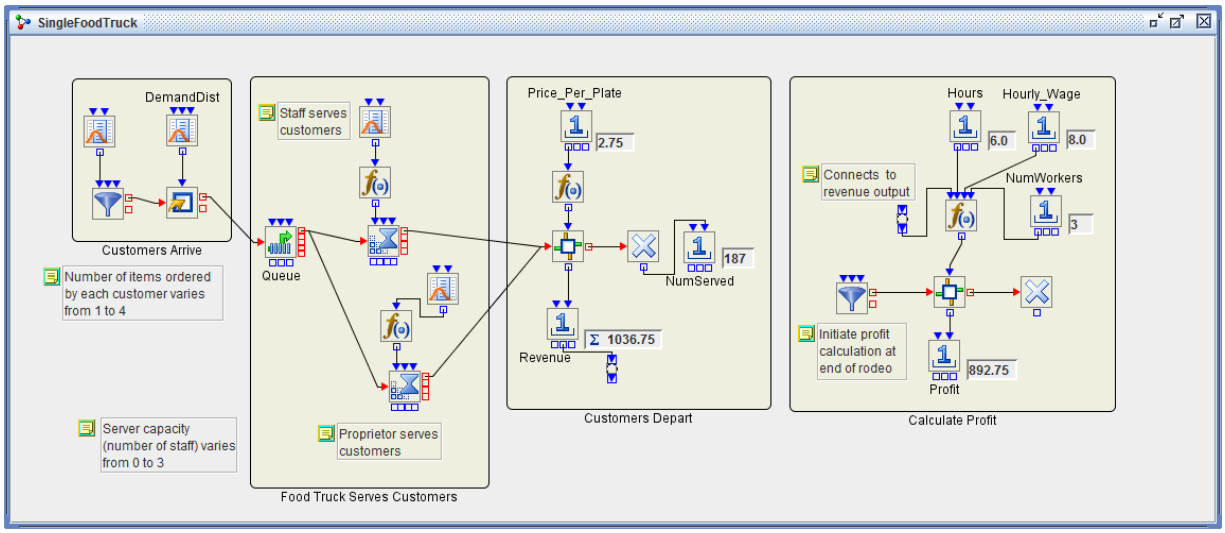

Figure 1. SAS Simulation Studio Food Truck Model

Customers arrive at a rate of 30 per hour, just as in the optimization model, but discrete-event simulation models the individual arrivals. Interarrival times are sampled from an exponential distribution with a mean of two minutes. The distribution of order sizes among customers is modeled directly by sampling from an empirical distribution that uses the figures shown in Table 1. Arriving customers enter a FIFO (first in, first out) queue and wait for service.

Preparation times for plates are exponentially distributed; the means are three minutes for the staff (20 per hour) and two minutes for the proprietor (30 per hour). There is no firm requirement that average hourly productivity be at least as high as average hourly demand, because discrete-event simulation models the individual orders and preparation times rather than aggregating. If productivity lags behind demand, the line grows and the model measures it. After receiving their food and paying, customers exit the model. The total profits are calculated at the end of the six-hour period, using revenue data gathered during the simulation run and input data on staffing costs.

The experiment for this model, shown in Figure 2, uses a full factorial experimental design that includes a design point for each unique combination of 4 prices and 4 staffing levels, or 16 design points in total.

SingleFoodTruckExp2

PointName	StartTime	EndTime	Hourly_Wage	Price_Per_Plate	NumWorkers	DemandDist	Replicates	Profit	NumServed	AvgQueueLength	MaxQueueLength	Avg_Staff_Util	Avg_Util_Prop
point 1	0	6	8	2	0	Y==pmf1	50	296	50	70.27610216940...	140	.	0.99740181...
point 2	0	6	8	2	1	Y==pmf1	50	553.08	99.86	37.30764534602...	77.14	0.98758226...	0.99336102...
point 3	0	6	8	2	2	Y==pmf1	50	728.24	137.3	19.16463107791...	41.82	0.96558067...	0.97808673...
point 4	0	6	8	2	3	Y==pmf1	50	830.12	162.48	6.479404414673...	18.76	0.87725288...	0.92008069...
point 5	0	6	8	2.25	0	Y==pmf2	50	401.625	66.32	55.15360084805...	110.72	.	0.99687824...
point 6	0	6	8	2.25	1	Y==pmf2	50	625.695	111.08	32.19823386922...	66.46	0.98539161...	0.98961119...
point 7	0	6	8	2.25	2	Y==pmf2	50	818.4	150.84	13.10545441109...	30.36	0.94096300...	0.95973480...
point 8	0	6	8	2.25	3	Y==pmf2	50	878.31	168.5	3.425508981179...	13.44	0.80767740...	0.87039150...
point 9	0	6	8	2.5	0	Y==pmf3	50	454	86.32	46.00613236469...	91.14	.	0.99452048...
point 10	0	6	8	2.5	1	Y==pmf3	50	688.9	140.06	18.83273231732...	40.34	0.96295373...	0.97166116...
point 11	0	6	8	2.5	2	Y==pmf3	50	786.6	168.12	4.066220236237...	15.44	0.81668991...	0.86919264...
point 12	0	6	8	2.5	3	Y==pmf3	50	771.45	174.32	0.843608227758...	7.84	0.62569852...	0.76198003...
point 13	0	6	8	2.75	0	Y==pmf4	50	502.755	91.56	43.87256357914...	86.98	.	0.99274970...
point 14	0	6	8	2.75	1	Y==pmf4	50	759.125	147.02	15.89405477373...	34.4	0.94842610...	0.95965120...
point 15	0	6	8	2.75	2	Y==pmf4	50	839.825	170.56	3.104793175380...	13.82	0.76797965...	0.83420315...
point 16	0	6	8	2.75	3	Y==pmf4	50	812.615	174.04	0.694929301128...	7.74	0.58436541...	0.74085586...

Figure 2. Experiment Window with Simulated Responses

Factors (control settings) in the experiment include the plate price and the staffing level, along with the hourly wage and the order size distribution that corresponds to the specified price. The experiment produces 50 replications of each design point and records several responses. Profit, which is also calculated by the optimization model, is recorded here. Other responses include the number of customers

served, the average and maximum length of the queue, and the average utilization of the staff and the proprietor. All responses are averaged over the specified 50 replications, and in SAS Simulation Studio you can expand any design point to display the responses from individual replications. These detailed data are also available for analysis. In the Experiment window, columns for factors are highlighted in yellow and response columns are highlighted in pink.

To maximize profit, the discrete-event simulation model agrees with the optimization model: the proprietor should set the price at $2.25 and have three workers on hand. Design point 8, corresponding to these choices, is highlighted in Figure 2. The profit associated with this decision is $878.31, much lower than the $949.50 profit predicted by the optimization model. This discrepancy emphasizes that assuming average behavior (in arrival rate and order size) over a long period can be somewhat misleading. Even though the discrete-event simulation model agrees with the recommendation of the optimization model, it provides a much more realistic view of the likely outcome. It also supplies queue and human resource utilization data. These are two elements that the optimization model necessarily ignores, owing to its aggregated view of system performance.

Because a full factorial design is used in this model, the simulated data give you a complete view of the response surfaces for multiple possible objectives: profit, customers served, average and maximum queue length, and average utilization of the proprietor and her staff. You can see which decisions optimize each of these metrics, and you can also consider the amount by which one set of decisions outperforms another set. The truck's proprietor might be willing to, for example, surrender a small amount of profit in order to reduce the maximum queue length or increase the number of customers she serves, either of which should improve her truck's reputation.

In this instance, the simulated data emphasize the broad appeal of the "$2.25 price, three workers" solution. It produces the greatest profit and is outperformed by only three other solutions in the number of customers served and average queue length. It produces the third-lowest maximum queue length. Staff utilization is above 80% and proprietor utilization exceeds 87%.

One alternative solution that is perhaps worth considering sets the price at $2.75 and uses two staff (plus the proprietor). Design point 15 in Figure 2 corresponds to this solution. It performs slightly better than the previous solution in the number of customers served and average queue length, performs very slightly worse in maximum queue length and labor utilization, and reduces the profit by only 4.4%. If the proprietor has trouble finding a third reliable worker, this would be an excellent alternative plan.

Outcome

By adding a discrete-event simulation view of the food truck pricing and staffing problem, you use information that an optimization approach struggles or fails to incorporate, and you produce metrics that optimization is unable to supply or acknowledge. This complements the profit-centric, aggregated prescriptive role of optimization by adding a detailed event-oriented and queueing-oriented perspective. Discrete-event simulation also emphasizes the importance of and interaction between the inherent variability in customer arrivals and preparation time. The combined optimization/simulation approach enables the proprietor to better understand how her decisions affect both profits and customer service performance. Balancing these two considerations will better enable her to improve her truck's standing in the marketplace.

EXAMPLE 2: WORK AREA OPTIMIZATION AT A EUROPEAN UTILITY

A major European utility company approached SAS with a request to assist in improving the assignment of its many thousands of field service engineers. The goal of the project was to determine the arrangement of work areas and the assignment of workers to them that would improve customer satisfaction and minimize worker travel time and total cost. This project is discussed in greater detail in a SAS® Global Forum 2014 paper (Yi, Lada, Smith, and Gray 2014).

Background

The territory served was divided into regions, which were further divided into subregions and then into work areas. Work areas consisted of groupings of contiguous postal sectors. The utility observed that service engineers spent approximately 10% of their time making service calls outside their assigned work

areas, causing an increase in travel time, overtime, and the use of subcontractors. Costs rose accordingly. The proposed solution was to find a better way to define work areas so that engineer skills and service calls would be better matched.

Optimization Models

SAS consultants generated a set of candidate work areas (individual groupings of postal sectors) and built a linear optimization model in PROC OPTMODEL to evaluate each candidate work area separately. There was considerable overlap in the coverage of postal sectors by candidate work areas. For each work area, a linear optimization matched predicted demand to engineers in the same area who possessed the required skills. The primary objective was to minimize unsatisfied demand, which was termed "shortage" and was measured in hours of unassigned work.

The individual linear optimizations produced an engineer-to-call assignment plan for each candidate work area. Next, one mixed integer linear optimization model, also built in PROC OPTMODEL, selected work areas from among the candidates to identify a final set of work areas that collectively covered all postal sectors in the subregion and were mutually exclusive—each postal sector belonged to exactly one work area. The primary objective of this phase of optimization, also, was to minimize shortage.

Discrete-Event Simulation Model

SAS consultants used SAS Simulation Studio to build the discrete-event simulation model, which simulated customer service activity in the subregion for six months, measured hourly. The most prominent input was the optimal work area configuration that corresponded to the optimal solution produced by the mixed integer linear optimization model. The simulation modeled the occurrence and disposition of each individual customer service call request within the time period, rather than assessing the entire stream of customer service calls as a group. For each call request, the model assigned an engineer and accounted for both the travel time of the engineer to the service location and the duration of the service call.

At the end of the simulation run, a SAS program produced response performance metrics. These were compared to the targeted service levels that were included (at an aggregated level) in the optimization model as constraints.

Outcome

Discrete-event simulation confirmed the predictions of the optimization model. Every customer service level statistic that was calculated from the simulated data met the stated requirements. This gave the utility additional confidence in the validity of the optimization approach.

The utility company could also have compared the simulated shortage figures to the optimal objective function value from the optimization model, but it chose to focus solely on fulfilling service-level requirements.

CONCLUSION

Both optimization and discrete-event simulation have proven to be invaluable aids in organizational planning and resource allocation for complex systems. Optimization provides critical decision guidance, and discrete-event simulation investigates system performance in multiple scenarios. Simulation modeling also incorporates important elements such as random variation and interaction, and adds an emphasis on the queueing aspects of a system. Together, the prescriptive elements of optimization and the predictive abilities of discrete-event simulation complement and reinforce each other to provide even greater benefits in planning and configuring systems and their resources. Discrete-event simulation can investigate the validity and accuracy of assumptions that are made during optimization modeling and can highlight elements of the optimization model that might need correction or adjustment.

In SAS/OR, the OPTMODEL modeling language in PROC OPTMODEL provides excellent and diverse optimization modeling and solution capabilities, and SAS Simulation Studio provides a powerful and intuitive graphical environment for building and executing discrete-event simulation models. In an increasing number of projects, they are being used together to guide SAS customers to more productive and more robust planning decisions.

REFERENCES

Lada, E. (2015a). "Simulate to Validate." *Operations Research with SAS* (blog). May 5. Available at http://blogs.sas.com/content/operations/2015/05/05/simulate-validate/.

Lada, E. (2015b). "Food Truck Analytics: Simulation or Queueing Model?" *Operations Research with SAS* (blog). October 13. Available at http://blogs.sas.com/content/operations/2015/10/13/food-truck-analytics-simulation-or-queueing-model/.

Yi, J., Lada, E., Smith, A., and Gray, C. (2014). "Work Area Optimization at a Major European Utility Company." In *Proceedings of the SAS Global Forum 2014 Conference.* Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENT

The authors thank Jack Rouse (SAS/OR software developer) for his assistance with elements of the optimization model for the food truck example.

RECOMMENDED READING

- *SAS Simulation Studio 14.1: User's Guide*
- *SAS/OR 14.1 User's Guide: Mathematical Programming*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Ed Hughes
SAS Institute Inc.
Ed.Hughes@sas.com

Emily Lada
SAS Institute Inc.
Emily.Lada@sas.com

APPENDIX: SAS CODE FOR THE FOOD TRUCK EXAMPLE OPTIMIZATION MODEL

```
/* Expected demand per customer for each price point */
data quant;
input case price expq;
cards;
1 2.00 3.0
2 2.25 2.7
3 2.50 2.1
4 2.75 2.0
;

proc optmodel;
/* The CASES set indexes the available price levels and related data */
set CASES;
num expquan{CASES};
num pricept{CASES};
```

```
/* Hours, hourly customer arrivals, hourly wage */
num hours=6;
num hrcusts=30;
num hrwage=8;

/* Hourly production by proprietor (p) and each staff member (s) */
num p_rate=30;
num s_rate=20;

/* Read the prices and corresponding expected order quantities */
read data quant into CASES=[case] pricept=price expquan=expq;

/* Set up the MILP optimization model */
var staff >=0 <=3 integer;
var setprice{CASES} binary;

/* Maximize profit */
max profit = sum{c in CASES}(setprice[c]*hours*hrcusts*expquan[c]*pricept[c])
             - (hours*hrwage*staff);

/* Meet expected hourly demand with hourly production */
con hrdemand: sum{c in CASES} (setprice[c]*expquan[c]*hrcusts) <= p_rate +
s_rate*staff;

/* Choose exactly one price */
con oneprice: sum{c in CASES} setprice[c] = 1;

/* Un-comment and use this command to fix a pricing decision */
/* Change the index of setprice as appropriate */
* fix setprice[1]=1;

/* Inspect the optimization model */
expand;

/* Invoke the MILP solver */
solve;

/* Output the optimal pricing and staffing decisions */
print pricept setprice staff;

quit;
```