Fuzzy Optimization and Normal Simulation for Solving Fuzzy Web Queuing System Problems

Xidong Zheng, Kevin Reilly
Dept. of Computer and Information Sciences
University of Alabama at Birmingham
Birmingham, AL, 35294, USA
205-934-2213
zhengx, reilly@cis.uab.edu

and

James J. Buckley
Department of Mathematics
University of Alabama at Birmingham
Birmingham, AL, 35294, USA
205-934-2154
buckley@math.uab.edu

ABSTRACT

In this paper, we use both fuzzy optimization and normal simulation methods to solve fuzzy web planning model problems, which are queuing system problems for designing web servers. We apply fuzzy probabilities to the queuing system models with customers arrival rate λ and servers' service rate μ, and then compute fuzzy system performance variables, including Utilization, Number (of requests) in the System, Throughput, and Response Time. For the fuzzy optimization method, we apply two-step calculation, first use fuzzy calculation to get the maximum and minimum values of fuzzy steady state probabilities, and then we compute the fuzzy system performance variables. For the simulation method, we use one-step normal queuing theory to simulate the whole system performance and its variables. We deal with queuing systems with a single server and multiple servers' cases, and compare the results of these two cases, giving a mathematical explanation of the difference.

Keywords: Fuzzy Optimization, Normal Simulation, Queuing Theory, and Web Planning Model.

1. INTRODUCTION

In this section, we introduce first fuzzy sets, fuzzy numbers, notations (section 1.1), and finally the queuing system models (section 1.2).

1.1 Fuzzy Notation

To indicate that a set is fuzzy, we place a bar over the symbol representing the fuzziness, as: $\overline{a_{ij}}$. If a set is known precisely, then we can denote it as: $\overline{a_{ij}} = a_{ij}$. Nevertheless, for simplicity and uniformity, we still write a_{ij} as $\overline{a_{ij}}$ in some contexts. So, $\overline{a}, \overline{x}, \overline{Y} \ldots$ all represent fuzzy sets. If $\overline{A}$ is a fuzzy set, we define $\overline{A}(x) \in [0,1]$ as the membership function of $\overline{A}$, for evaluating a real number x. And a α-cut of $\overline{A}$, written as $\overline{A}[\alpha]$, is defined as $\{x| \overline{A}(x) \geq \alpha\}$, for $0 \leq \alpha \leq 1$. A fuzzy number $\overline{N}$ is a fuzzy subset of the real numbers satisfying:

1) $\overline{N}(x) = 1$ for some x (normalized); and
2) $\overline{N}[a]$ is a closed, bounded, interval, for all $0 \leq \alpha \leq 1$.

A triangle fuzzy number $\overline{T} = (a_1 / a_2 / a_3)$ is defined by three numbers $a_1 < a_2 < a_3$, where the graph of $y = \overline{T}(x)$ is a triangle with base on the interval $[a_1, a_3]$ and vertex at x= a_2 ($\overline{T}(a_2) = 1$). All the fuzzy numbers we used in this paper are triangle fuzzy numbers [1][2][3].

1.2 Queuing Models

From traditional queuing system, we know that it has customers' arrival rate with Poisson probability distribution, which means that there is a positive constant λ so that the probability of m arrivals of customers per unit time is $\lambda^m * e^{(-\lambda)} / m!$. And the servers' service time is exponential distributed with service rate μ.

In our model, we regard the web system as a finite-queuing system, the Internet users as the customers, and web servers as the servers [4][5]. The website accepts at most M number of users' requests at a time, where M is the number of users in the system and in the queue. To apply the fuzzy probability in our system, we get that the customers' arrival rate is $\overline{\lambda}$ and servers' service rate is $\overline{\mu}$, then to compute the fuzzified system performance variables: Utilization ($\overline{U}$), Number (of requests) in the System ($\overline{N}$), Throughput ($\overline{X}$), and Response Time ($\overline{R}$). To simplify our computation, we will compute the one-server case and two-server case with the capacity of accepting M= 4 users in one time at most.

2. WEB PLANNING MODELS AND SYSTEM PERFORMANCE

In this section, we describe the system models of the one-server case and two-server case and their characteristics, system performance variables formula according to arrival rate and service rate.

2.1 Single-Server Case

In single-server case, the whole system has only one web server. We assume that the infinite number of users around the world visit the website with average arrival rate of λ (Poisson arrival), and the average service rate of μ (exponential service time) (see Figure 1). If the number of users arriving at the same time exceeds the limit of the system buffer, the users will get "server is busy" response and leave the system.

2.1.1 Markov Chain for Single-Server case: We regard the single server system as a Markov chain (see Figure 2) and assume that:

- State k → k customers in the system;
- P (I, J) → Probability of transition from state of I to state of J.

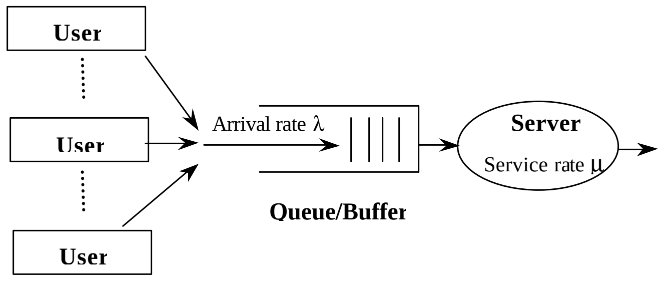

Figure1. Single-Server queuing model

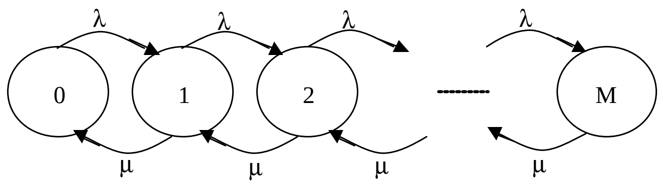

Figure 2. State transition diagram — Infinite users/finite queue/single server

2.1.2 Property of the single-server Markov chain: Assuming that the internal time period δ is small enough, i.e. in the limit as δ →0, we get the state transition probability expressions [6]:

- P (0, 0) = 1 - λ; P (M, M) =1 - μ;
- P (j, j+1) = λ; (0 <= j <M)
- P (j, j) = 1 - λ - μ; (0 < j < M)
- P (j, j-1) = μ; (0 < j <=M)
- P (i, j) = 0 (for all other i, j).

It is actually a tri-diagonal Markov chain matrix (see Figure 3).

So, in the limit as δ → 0, transitions exist only between adjacent States. And λ, μ are flow rates between states.

$$\begin{bmatrix} 1-\lambda & \lambda & 0 & \cdots & 0 \\ \mu & 1-\lambda-\mu & \lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \mu & 1-\lambda-\mu & \lambda \\ 0 & \cdots & 0 & \mu & 1-\mu \end{bmatrix}$$

Figure 3. Tri-diagonal Markov chain matrix (single-server)

2.1.3 Equilibrium analysis of single-server system: We want to obtain the probability of being in state k, denoted as: P (k).

At equilibrium, we have local balance equations between two states (k, k+1)[6]:

λ * P (k) = μ * P (k+1), (for all k)

Let ρ = λ/μ, we get:

P (k)= ρ^k * P (0), and $\sum_{k=0}^{M} P(k) = 1$. Thus:

Probability of being in state k:

$$P(k) = \frac{1-\rho}{1-\rho^{M+1}} * \rho^k \quad k=0, 1, \ldots, M \quad \ldots\ldots\ldots\ldots\ldots\ldots(2.1\text{-}1)$$

Server utilization:

$$U = 1 - P(0) = \frac{\rho * [1-\rho^M]}{1-\rho^{M+1}} \ldots\ldots\ldots\ldots\ldots\ldots(2.1\text{-}2)$$

Average number of requests in the server:

$$N = \sum_{k=0}^{M} k * P(k) = \frac{\rho * [M * \rho^{M+1} - (M+1) * \rho^M + 1]}{(1-\rho^{M+1}) * (1-\rho)} \ldots\ldots(2.1\text{-}3)$$

Average server throughput:

$$X = U * \mu = \frac{\lambda * (1-\rho^M)}{1-\rho^{M+1}} \ldots\ldots\ldots\ldots\ldots\ldots(2.1\text{-}4)$$

Average response time:

$$R = N / X = \frac{M * \rho^{M+1} - (M+1) * \rho^M + 1}{\mu * (1-\rho^M) * (1-\rho)} \ldots\ldots\ldots\ldots(2.1\text{-}5)$$

2.2 Multiple-Server Case

In multiple-server case, the whole system has C parallel and identical web servers. We still assume that the infinite number of users around the world visit the website with average arrival rate of λ (Poisson arrival), and all the servers' average service rate are μ (exponential service time) (see Figure 4). Also, if the number of users arriving at the same time exceeds the limit of the system buffer, the users will then get "servers are busy" response and leave the system.

2.2.1 Markov Chain for multiple-Server case: The multiple-server case Markov chain is similar to the single server case (see Figure 5). Still we assume that:

- State k → k customers in the whole system (including all servers);
- P (I, J) → Probability of transition from state of I to state of J.

But here the difference is that departure rate is proportional to the number of servers in use.

2.2.2 Property of the multiple-server Markov chain: Similar to single-server case, we assume that the internal time period δ is small enough, i.e. $\delta \rightarrow 0$, we can get state transition probability like these:

- $P(0, 0) = 1 - \lambda$; $P(M, M) = 1 - C\mu$;
- $P(j, j+1) = \lambda$ (for $0 <= j < M$);
- $P(j, j-1) = j * \mu$ (for $0 < j <= C$);
- $P(j, j) = 1 - \lambda - j * \mu$ (for $0 < j <= C$);
- $P(j, j-1) = C * \mu$ (for $C < j <= M$);
- $P(j, j) = 1 - \lambda - C * \mu$ (for $C < j < M$);
- $P(i, j) = 0$ (for all other i, j).

Transitions again occur only between adjacent states and the transition matrix is still a tri-diagonal Markov chain matrix (see Figure 6).

2.2.3 Equilibrium analysis of multi-server system: Similar to single-server case, we denote the probability of being in state k as: $P(k)$. At equilibrium, the balance equations between two states (k-1, k) become:

- $\lambda * P(k-1) = k * \mu * P(k)$ (for $0 < k <= C$);
- $\lambda * P(k-1) = C * \mu * P(k)$ (for $C < k <= M$);

And again let $\rho = \lambda/\mu$, we get:

Probability of being in state k:

$$P(k) = \begin{cases} P(0) * \rho^{k} / k! & \text{(For } 0 <= k <= C\text{)} \\ P(0) * C^{C-k} * \rho^{k} / C! & \text{(For } C < k <= M\text{)} \end{cases} \quad \text{….(2.2-1)}$$

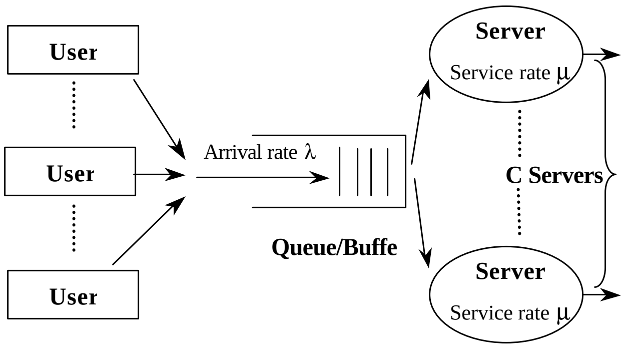

Figure 4. Multiple-Server queuing model

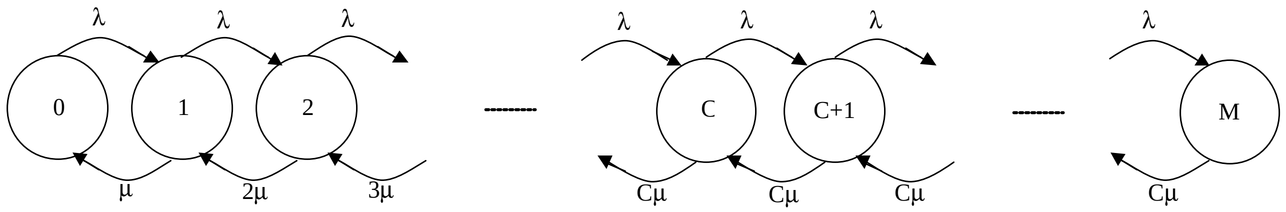

Figure 5. State transition diagram —Infinite users/finite queue/multiple servers

$$\begin{array}{c|cccccccccc|c}
 & & & & \text{Column C} & & & & & & & \\
 & 1-\lambda & \lambda & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & \\
 & \mu & 1-\lambda-\mu & \lambda & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & \\
 & 0 & 2\mu & 1-\lambda-2\mu & \lambda & 0 & \cdots & \cdots & \cdots & \cdots & 0 & \\
 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\
\text{Row C} & 0 & \cdots & C\mu & 1-\lambda-C\mu & \lambda & 0 & \cdots & \cdots & \cdots & 0 & \text{Row C} \\
 & 0 & \cdots & 0 & C\mu & 1-\lambda-C\mu & \lambda & 0 & \cdots & \cdots & 0 & \\
 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\
 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & C\mu & 1-\lambda-C\mu & \lambda & 0 & \\
 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & C\mu & 1-\lambda-C\mu & \lambda & \\
 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & C\mu & 1-C\mu & \\
 & & & & \text{Column C} & & & & & & &
\end{array}$$

Figure 6. Tri-diagonal Markov chain matrix (multiple servers)

Using $\sum_{k=0}^{M} P(k) = 1$, we get:

$$P(0) = \left[\sum_{k=0}^{C-1} \frac{\rho^k}{k!} + \frac{\rho^C * (C^{M-C+1} - \rho^{M-C+1})}{C! * C^{M-C} * (C-\rho)}\right]^{-1} \quad \ldots\ldots\ldots\ldots(2.2\text{-}2)$$

Server utilization:
$U = 1 - P(0)$ ……………………………………………(2.2-3)

Average number of requests in the server:

$$N = \sum_{k=0}^{M} k * P(k) \quad \ldots\ldots\ldots\ldots(2.2\text{-}4)$$

Average server throughput:

$$X = \sum_{k=0}^{C-1} k * \mu * P(k) + C * \mu * \sum_{k=C}^{M} P(k) \quad \ldots\ldots\ldots\ldots(2.2\text{-}5)$$

Average response time:
$R = N / X$ ……………………………………………(2.2-6)

3. FUZZY OPTIMIZATION OF THE WEB SYSTEM

Up to now, the system is a crisp one. In reality, people often describe the user arrival rate and server service time in a fuzzy language. So we want to apply the fuzzy probability into our system [1][2][3], fuzzify the arrival rate λ and service rate μ and denote them as $\overline{\lambda}$ and $\overline{\mu}$ in respect, then we use the fuzzy calculation theory to compute the fuzzified system performance variables: Utilization ($\overline{U}$), Number (of requests) in the System ($\overline{N}$), Throughput ($\overline{X}$), and Response Time ($\overline{R}$).

For fuzzy calculation in our problem, we have the following two methods:

- One-step method. We begin with the crisp formula of U, N, X, R, which are directly the function of λ and μ and substitute values from M and C into them to get expressions from which the maximum and minimum value of U, N, X, R can be determined and utilized as the final fuzzy result of $\overline{U}$, $\overline{N}$, $\overline{X}$ and $\overline{R}$.
- Two-step method. First, we compute the fuzzy steady state probability $\overline{P(k)}$ according to $\overline{\lambda}$ and $\overline{\mu}$. Then using the crisp formulae for U, N, X, R as a function of P (k), but now employing fuzzy arithmetic, we compute fuzzy results, $\overline{U}$, $\overline{N}$, $\overline{X}$ and $\overline{R}$.

3.1 One-server example

For illustration purposes, we let M=4 and fuzzy number $\overline{\lambda}$ =(3 / 4 / 5), $\overline{\mu}$ = (5 / 6 / 7). According to formulas 2.1-1 to 2.1-5, we have done the following two cases fuzzy optimization:

3.1.1 One-Step optimization: Using the direct function of λ and μ, the formula of U, N, X, R are ($\rho = \lambda/\mu$):

$$U = \frac{\rho[1-\rho^4]}{1-\rho^5}; \qquad N = \frac{\rho[4\rho^5 - 5\rho^4 + 1]}{(1-\rho 5)(1-\rho)};$$

$$X = \frac{\lambda(1-\rho^4)}{1-\rho^5}; \qquad R = \frac{4\rho^5 - 5\rho^4 + 1}{\mu(1-\rho^4)(1-\rho)}.$$

A direct, simple computation of $\overline{U}$, $\overline{N}$, $\overline{X}$ and $\overline{R}$ (alpha zero cut) can be effected in MATLAB optimization toolbox. The results appear in see Table 1.

3.1.2 Two-step optimization: Unlike one-step method, we first compute the fuzzy steady state probability $\overline{P(k)}$ according to the fuzzy number $\overline{\lambda}$ =(3 / 4 / 5), $\overline{\mu}$ = (5 / 6 / 7). From equation 2.1-1 and M=4, we know that:

$$P(k) = \frac{1-\rho}{1-\rho^5} * \rho^k \qquad k=0, 1, \ldots, 4$$

The fuzzy results for these fuzzy probabilities are in Table 2.

After obtaining fuzzy steady state probabilities, we next need to compute the (final) fuzzy system performance variables $\overline{U}$, $\overline{N}$, $\overline{X}$ and $\overline{R}$.

$$\overline{U}[0] = [\min(\sum_{k=1}^{4} p(k)), \max(\sum_{k=1}^{4} p(k))];$$

$$\overline{N}[0] = [\min(\sum_{k=1}^{4} k * P(k)), \max(\sum_{k=1}^{4} k * P(k))];$$

$$\overline{X}[0] = [\min(\sum_{k=1}^{4} \mu * P(k)), \max(\sum_{k=1}^{4} \mu * P(k))];$$

$$\overline{R}[0] = [\min(\sum_{k=1}^{4} k * P(k) / \sum_{k=1}^{4} \mu * P(k)),$$
$$\max(\sum_{k=1}^{4} k * P(k) / \sum_{k=1}^{4} \mu * P(k))],$$

All these object functions are subject to the constraints:
$P(k) \in \overline{P(k)}[0]$, $0 <= k <= 4$, and $\sum_{k=0}^{4} P(k) = 1$;

Using MATLAB, we get the optimization result in Table 1.

Table 1. Fuzzy system performance variables (single-server; methods, one- and two-step). See Table 2 for intermediate steps in the two-step case.

	Alpha Zero Cut	
	One-step method	**Two-step method**
$\overline{U}$	[0.4202, 0.8000]	[0.4202, 0.8000]
$\overline{N}$	[0.6767, 2.0000]	[0.6767, 2.0000]
$\overline{X}$	[2.8314, 4.5432]	[2.1010, 5.6000]
$\overline{R}$	[0.2300,0.5000]	[0.1208, 0.9519]

3.2 Two-server example

Like the one-server case, in order to exemplify the computation, we again let M=4 and employ fuzzy arrival and service rates, $\overline{\lambda}$ =(3 / 4 / 5), $\overline{\mu}$ = (5 / 6 / 7). According to formulas 2.2-1 to 2.2.6, we've computed both one-step and two-step fuzzy optimizations in the manner we now describe.

Table 2. Fuzzy steady state probability (two-step method) – intermediate value preceding final values in Table 1.

	Alpha Zero Cut	
	Single-server	**Two-server**
$\overline{P(0)}$	[0.2000,0.5798]	[0.3478, 0.6475]
$\overline{P(1)}$	[0.2000,0.2608]	[0.2775, 0.3503]
$\overline{P(2)}$	[0.1065,0.2000]	[0.0595, 0.1739]
$\overline{P(3)}$	[0.0456,0.2000]	[0.0127, 0.0870]
$\overline{P(4)}$	[0.0196,0.2000]	[0.0027, 0.0435]

3.2.1 One-Step optimization: Using the expressions for λ and μ, the formula of U, N, X, R are ($\rho = \lambda/\mu$):

$$U = 1-\left[1+\rho+\frac{\rho^2*(8-\rho^3)}{8(2-\rho)}\right]^{-1};$$

$$N = (\rho+\rho^2+3\rho^3/4+\rho^4/2)/\left(1+\rho+\frac{\rho^2*(8-\rho^3)}{8(2-\rho)}\right);$$

$$X = \mu*(\rho+\rho^2+\rho^3/2+\rho^4/4)/\left(1+\rho+\frac{\rho^2*(8-\rho^3)}{8(2-\rho)}\right);$$

$$R = \frac{(\rho+\rho^2+3\rho^3/4+\rho^4/2)}{\mu*(\rho+\rho^2+\rho^3/2+\rho^4/4)}.$$

Using MATLAB, we get the alpha zero cut of $\overline{U}$, $\overline{N}$, $\overline{X}$ and $\overline{R}$ (see Table 3 below).

3.2.2 Two-step optimization: From equation 2.2-1 and M=4, we get:

$$P(0) = \left[1+\rho+\frac{\rho^2*(8-\rho^3)}{8(2-\rho)}\right]^{-1};$$

$$P(1) = \rho*\left[1+\rho+\frac{\rho^2*(8-\rho^3)}{8(2-\rho)}\right]^{-1};$$

$$P(2) = \frac{\rho^2}{2}*\left[1+\rho+\frac{\rho^2*(8-\rho^3)}{8(2-\rho)}\right]^{-1};$$

$$P(3) = \frac{\rho^3}{4}*\left[1+\rho+\frac{\rho^2*(8-\rho^3)}{8(2-\rho)}\right]^{-1};$$

$$P(4) = \frac{\rho^4}{8}*\left[1+\rho+\frac{\rho^2*(8-\rho^3)}{8(2-\rho)}\right]^{-1}.$$

Then we compute the fuzzy steady state probability $\overline{P(k)}$ according to the fuzzy numbers $\overline{\lambda} = (3 / 4 / 5)$, $\overline{\mu} = (5 / 6 / 7)$ (see Table 2).

Then similar to one-server case, we compute the fuzzy system performance variables $\overline{U}$, $\overline{N}$, $\overline{X}$ and $\overline{R}$ as following:

$$\overline{U}[0] = [\min(\sum_{k=1}^{4} p(k)), \max(\sum_{k=1}^{4} p(k))];$$

$$\overline{N}[0] = [\min(\sum_{k=1}^{4} k*P(k)), \max(\sum_{k=1}^{4} k*P(k))];$$

$$\overline{X}[0] = [\min(\mu * P(1) + 2\mu * \sum_{k=2}^{4} P(k)),$$
$$\max(\mu*P(1)+2\mu*\sum_{k=2}^{4} P(k))];$$

$$\overline{R}[0]=[\min(\sum_{k=1}^{4} k*P(k)/(\mu*P(1)+2\mu*\sum_{k=2}^{4} P(k))),$$
$$\max(\sum_{k=1}^{4} k*P(k)/(\mu*P(1)+2\mu*\sum_{k=2}^{4} P(k)))],$$

All these object functions are subject to the constraints:

$P(k) \in \overline{P(k)}[0]$, $0 <= k <= 4$, and $\sum_{k=0}^{4} P(k) = 1$;

Using MATLAB, we get the optimization result in Table 3.

Table 3. Fuzzy system performance variables (two servers; methods, one- and two-step). See Table 2 (again) for intermediate steps for the two-step case.

	Alpha Zero Cut	
	One-step method	**Two-step method**
$\overline{U}$	[0.3525, 0.6522]	[0.3525, 0.6522]
$\overline{N}$	[0.4456, 1.1304]	[0.4456, 1.1304]
$\overline{X}$	[2.9737, 4.9223]	[2.1370, 6.6962]
$\overline{R}$	[0.1489, 0.2364]	[0.0665, 0.5291]

3.3 Comparison of One-step and Two-step Results

From the computation results of both one- and two-server cases, we see that the fuzzy utilization and number of users requests are the same in both one-step and two-step methods. Meanwhile, the system throughput and average response time are broader in two-step than in one-step method.

There are two possible reasons related to the computation results:

- From the equations of system performance variables, we can see that U and N are the function of only the ratio (ρ) of user arrival rate (λ) and server service rate (μ); but X and R are the function of ρ and λ or μ;
- In two-step method, we first compute the fuzzy steady state probability, then the fuzzy system performance variables. Thus, this computation has an additional fuzzy step, which results in increased and expanded fuzzy spread (in both left and right sides of the fuzzy numbers).

4. Simulation method

We have also implemented another approach dealing with these fuzzy problems, i.e. a standard (stochastic) simulation. Unlike fuzzy optimization, simulation is based on traditional queuing models, using statistical and experimental methods to generate an internal picture of the system from which we gather the (statistical) data for analyzing system performance variables [7][8].

4.1 Simulation Software

Several software packages can be found for simulations purpose. We use the widely known simulation languages, GPSS-H and SLX (an Object Based language which also includes the heart of GPSS-

H) [8]. SLX stands for Simulation Language with eXtensibility, it is a well-conceived, layered simulation system. Users of the upper layers can ignore lower layers.

Simulation languages such as GPSS and SLX have built-in probability functions that we can directly employ in our problem, e.g., exponential functions for inter-arrival and service times. Random number generators can be assigned to these functions to allow appropriate replication capabilities. On the output side frequency distributions often are generated automatically and graphs of them are easy to obtain, i.e., with only a few statements.

Model strategy and subsequent program development for our web planning models are easy to incorporate into these software packages by high-level constructs. The Poisson arrival and service rates are rendered in terms of exponential inter-arrival intervals and service time distributions, compatible with the mathematics of our situations and readily defined in the language.

Also in these languages, there are a variety of blocks or statements, which are conjugates of each other, defining the starting and ending events of an activity: joining a queue (in SLX, e.g.) is represented by enqueue block with the user-defined queue name and leaving is represented by a depart block with the name (See Figure 7).

```
Arrivals: customer
      iat = rv_expo (random-stream1, arrival rate)
      util_time = stop_time;
Enqueue queue;
Seize (Enter) server(s);
Departure queue;
Advance rv_expo (random-stream2, service rate);
Release (Leave) server(s);
```

Figure 7. Some statements structures of SLX

4.2 Simulation examples

According to the same general plan as above, we use SLX to implement simulation to obtain fuzzy results for Utilization ($\overline{U}$), Number (of requests) in the System ($\overline{N}$), Throughput ($\overline{X}$), and Response Time ($\overline{R}$). Again, we let M=4 and employ fuzzy number $\overline{\lambda}$ = (3 / 4 / 5), $\overline{\mu}$ = (5 / 6 / 7) in illustrations.

In simulation, we do not need to simulate all combinations of fuzzy number of $\overline{\lambda}$ and $\overline{\mu}$ to get some of our desired results. We just use the middle points of arrival rate and service rate to tackle the problems. We start the simulation using the structure in figure 7; repeat the simulation for certain number of loops. We then tabulate all the statistical data of server **Utilization (U)**, average **Number of requests (N)** in the system, system **Throughput (X),** and average **Response Time (R).** Through analysis, we get the maximum and minimum of all four variables, along with their mean values. (See Table 4). Also, the (entire) distribution curves of U, N, X, R are made available. Additional variations of these themes produce additional output values and the insights acquired through their interpretations enrich the overall computational scheme.

Table 4. Fuzzy system performance variables (Simulation)

	Alpha Zero Cut	
	Single-Server	**Two-Server**
$\overline{U}$	[0.4114, 0.7934]	[0.3592, 0.6472]
$\overline{N}$	[0.6617, 1.9846]	[0.4494, 1.0914]
$\overline{X}$	[2.8333, 4.5417]	[2.9583, 4.9983]
$\overline{R}$	[0.2277, 0.5016]	[0.1406, 0.2397]

4.3 Advantages of using simulation

Comparing the simulation results to those of fuzzy optimization, in the one-step optimization case (both one- and two-server situations) we see that there is a very good match. There are some decided advantages in using simulation over the fuzzy computation [8].

- Simulation itself is more natural than fuzzy computation in the sense of queuing model theory. It provides not only the correct results, but also tells how the queuing system works in reality;
- Simulation languages can provide much automatic output from their statistics running data and can provides more statistics that fuzzy computation cannot get, like data distributions;
- Simulation languages have a set of built-in files and lists that are used to store an agenda (a list of future events). One or more lists are used to keep track of dynamic (movable simulation) entities that, at any given moment, are ready to move but, say, are blocked from moving, as in the case of entities that would enter a server were it available. The languages readily handle several complexities, e.g., balking and preemption, which we may utilize in our future studies.

5. CONCLUSIONS

We have first presented fuzzy optimization methods to solve web planning queuing problems. In fuzzy optimization, we presented both a one- and a two-step computational method. In the former case, fuzzy number is introduced after conventional argument up to the system performance values themselves. In the latter case, we compute fuzzy system steady state probabilities as an intermediate step and then proceed, via fuzzy arithmetic, to the final results. The latter method produces results with a broader fuzzy spread because of the additional fuzzy arithmetic.

Also, we have considered both singer-server and multiple-server cases and gave a detailed description of the system performance variables in both cases. And we have illustrated some typical examples to make procedures easier to understand.

Finally, we covered a conventional simulation based method to solve fuzzy problems, which may be a "first" in the fuzzy probability field. Using examples in both single- and two-server cases, we demonstrated that conventional simulation is appropriate, and effective, in fuzzy probability modeling. In some instances, such as developing (output) data distributions, it seems to have proven to be better.

6. REFERENCES

[1] J.J. Buckley, Kevin Reilly, and Xidong Zheng, Fuzzy Probabilities for Web Planning, Soft Computing, In press.

[2] J.J. Buckley, Fuzzy Probabilities and Fuzzy Sets for Web Planning. Physica-Verlag, Heidelberg, Germany, 2004.

[3] J. J. Buckley, Fuzzy Probabilities: New Approach and Applications, Physica-Verlag, Heidelberg, 2002.

[4] Xidong Zheng, K. Reilly, J. J. Buckley, Comparing Genetic Algorithms And Exhaustive Methods Used In Optimization Problems For Fuzzy Probability-Based Web Planning Models. In Proc. The International Conference On Artificial Intelligence, Las Vegas Nevada, USA, June 2003,pp.463-468.

[5] Xidong Zheng, K. Reilly, J. J. Buckley, Applying Genetic Algorithms To Fuzzy Probability-Based Web Planning Models. In Proc. ACMSE'03, Savanna, Georgia, USA, March 2003, pp.241-245.

[6] D. A. Menase and V. A. F. Almeida, Capacity Planning for Web Performance: Metrics, Models and Methods, Prentice-Hall.

[7] K. Reilly, J. J. Buckley, X. Zheng, and F. Hernandez, Monte Carlo, Statistics, and Fuzzy Uncertain Probabilities. In Proc. 2002 Huntsville Simulation Conference, San Diego, CA, 2002. Society for Computer Simulation, Int'l. 6 pp. In press.

[8] J.O.Henriksen and R.C.Crain. GPSS/H Reference Manual, 4th edition. Wolverine Software, Inc., Alexandria, VA, 1995.