Making Decisions via Simulation

Ref: Law and Kelton, Chapter 10; Handbook of Simulation, Chapters 8 and 9; Haas, Sec. 6.3.6.

We give an introduction into some methods for selecting the “best” system design (or parameter setting for a given design), where the performance under each alternative is estimated via simulation.

The methods presented are chosen because of simplicity—see the references for more details on exactly when they are applicable, and whether improved versions of the algorithms are available. Current understanding of the behavior of these algorithms is incomplete, and they should be applied with due caution.

1. Continuous Stochastic Optimization

Robbins-Monro Algorithm

Here we will consider the problem $\min_{\theta \in \Theta} f(\theta)$, where, for a given value of $\theta$, we are not able to evaluate $f(\theta)$ analytically or numerically, but must obtain (noisy) estimates of $f(\theta)$ using simulation. We will assume for now that the set of possible solutions $\Theta$ is uncountably infinite. In particular, suppose that $\Theta$ is an interval $[\theta, \theta]$ of real numbers. One approach to solving the problem $\min_{\theta \in \Theta} f(\theta)$ is to estimate $f'(\theta)$ using simulation and then use an iterative method to solve the equation $f'(\theta) = 0$. The most basic such method is called the Robbins-Monro Algorithm, and can be viewed as a stochastic version of Newton’s method for solving nonlinear equations.

The basic iteration is

$$\theta_{n+1} = \pi\left(\theta_n - \left(\frac{a}{n}\right) Z_n\right),$$

where $a > 0$ is a fixed parameter of the algorithm (the quantity $a/n$ is sometimes called the “gain”), $Z_n$ is an unbiased estimator of $f'(\theta_n)$, and

$$\pi(\theta) = \begin{cases} 0 & \text{if } \theta < \theta; \\ \theta & \text{if } \theta \leq \theta \leq \theta; \\ \theta & \text{if } \theta > \theta. \end{cases}$$

(The function $\pi(\cdot)$ projects the current parameter value onto the feasible set. Algorithms that iterate as described above are called stochastic approximation algorithms.) Denote by $\theta^*$ the global minimizer of $f$. If the only local minimizer of $f$ is $\theta^*$, then under very general conditions

$$\theta_n \to \theta^*$$
as \( n \to \infty \) with probability 1. (Otherwise the algorithm can converge to some other local minimizer.) For large \( n \), it can be shown that \( \theta_n \) has approximately a normal distribution. Thus, if the procedure is run \( m \) independent times (where \( m = 5 \) to 10) with \( n \) iterations for each run, generating i.i.d. replicates \( \theta_{n,1}, \ldots, \theta_{n,m} \), then a point estimator for \( \theta^* \) is 

\[
\hat{\theta}_m = \frac{1}{m} \sum_{j=1}^{m} \theta_{n,j}
\]

and confidence intervals can be formed in the usual way, based on the Student-t distribution with \( m - 1 \) degrees of freedom.

Remark: Variants of the above procedure are available for multi-parameter problems.

Remark: The main drawback of the above algorithm is that convergence can be slow and sensitive to the choice of the parameter \( a \); current research is focused on finding methods that converge faster. One easy, practical way to stabilize the algorithm a bit is to keep track and return the best parameter value seen so far.

Remark: A variant of the Robbins-Monro procedure, due to Kiefer and Wolfowitz, replaces the gradient estimate by a finite-difference that is obtained by running the simulation at various values of the parameter \( \theta \). This procedure breaks down as the dimension of the parameter space increases. Recent work on “simultaneous perturbation stochastic approximation” (SPSA) tries to avoid the dimensionality problem as follows. At the \( k^{th} \) iteration of a \( d \)-dimensional problem, a finite-difference is obtained by running the simulation at the two parameter values \( \theta \pm c\Delta_k \). Here \( \theta \) is the current value of the parameter vector (of length \( d \)), \( c \) is a nonnegative real number that is a parameter of the algorithm, and \( \Delta_k \) is a \( d \)-vector of independent Bernoulli random variables, each equal to 1 or –1 with equal probability.

How do we obtain the derivative estimates needed for the Robbins-Monro algorithm? One popular method for estimating \( f'(\theta) \) is via likelihood ratios.

Tutorial on Likelihood Ratios

Suppose that we wish to estimate \( \alpha = E[g_n(X_0, X_1, \ldots, X_n)] \), where \( X_0, X_1, \ldots, X_n \) are i.i.d. replicates of a discrete random variable \( X \) with state space \( S \). The straightforward approach would be as follows: for \( k = 1, 2, \ldots, K \), generate \( n+1 \) i.i.d. replicates of \( X \), say \( X_{0,k}, X_{1,k}, \ldots, X_{n,k} \), and set

\[
\alpha(k) = g_n(X_{0,k}, X_{1,k}, \ldots, X_{n,k}).
\]

Then estimate \( \alpha \) by the average of \( \alpha(1), \ldots, \alpha(K) \).

Suppose that \( Y \) is another random variable with state space \( S \). Also suppose for all \( s \in S \) that \( q(s) = P\{Y = s\} \) is positive whenever \( p(s) = P\{X = s\} \) is positive. Then \( \alpha \) can be estimated based on simulation of \( Y \) instead of \( X \). Let \( Y_0, \ldots, Y_n \) be i.i.d. replicates of \( Y \), and set
\[ L_n = \frac{\prod_{i=0}^{n} p(Y_i)}{\prod_{i=0}^{n} q(Y_i)}. \]

\( L_n \) is called a likelihood ratio and represents the likelihood of seeing the observations under the distribution \( p \) relative to the likelihood of seeing the observations under distribution \( q \). We claim that

\[ \mathbb{E}[g_n(Y_0, Y_1, \ldots, Y_n)L_n] = \mathbb{E}[g_n(X_0, X_1, \ldots, X_n)] \]

To see this, observe that

\[ \mathbb{E}[g_n(Y_0, Y_1, \ldots, Y_n)L_n] = \sum_{s_0 \in S} \cdots \sum_{s_n \in S} g_n(s_0, \ldots, s_n) \left( \prod_{i=0}^{n} p(s_i) \right) \prod_{i=0}^{n} q(s_i) \]

\[ = \sum_{s_0 \in S} \cdots \sum_{s_n \in S} g_n(s_0, \ldots, s_n) \left( \prod_{i=0}^{n} p(s_i) \right) \prod_{i=0}^{n} q(s_i) \]

\[ = \mathbb{E}[g_n(X_0, \ldots, X_n)] \]

Thus, we can simulate \( Y \)'s instead of \( X \)'s, as long as the performance measure \( g \) is multiplied by the likelihood ratio “correction factor.”

This idea can be applied in great generality. For example, if \( X \) and \( Y \) are real-valued random variables with density functions \( f_X \) and \( f_Y \) respectively, then (1) holds with

\[ L_n = \frac{\prod_{i=0}^{n} f_X(Y_i)}{\prod_{i=0}^{n} f_Y(Y_i)}. \]

As another example, if \( X_0, X_1, \ldots, X_n \) are observations of a discrete-time Markov chain with initial distribution \( \mu \) and transition matrix \( P \) and \( Y_0, Y_1, \ldots, Y_n \) are observations of a discrete-time Markov chain with initial distribution \( \mu' \) and transition matrix \( \bar{P} \), then (1) holds with
\[ L_n = \frac{\mu(Y_0) \prod_{i=1}^{n} P(Y_{i-1}, Y_i)}{\tilde{\mu}(Y_0) \prod_{i=1}^{n} \tilde{P}(Y_{i-1}, Y_i)}. \]

**Remark:** The relation in (1) continues to hold if \( n \) is replaced by a random time \( N \).

**Remark:** \( L_n \) can easily be computed in a recursive manner.

Likelihood ratios can be computed in the context of GSMPs in an analogous way. The idea is to initially set the “running” likelihood ratio equal to 1. Then, whenever there is a state transition from \( S_n \) to \( S_{n+1} \) caused by the occurrence of the events in \( E_n^* \), the running likelihood ratio is multiplied by \( p(S_{n+1}; S_n, E_n^*)/p(S_{n+1}; S_n, E_n) \). Moreover, for each new clock reading \( X_i \) generated for an event \( e_i \), the running likelihood is multiplied by \( f(X_i; S_{n+1}, e_i, S_n, E_n^*)/\tilde{f}(X_i; S_{n+1}, e_i, S_n, E_n^*) \). (Here \( f \) and \( \tilde{f} \) denote the densities of the new clock-setting distributions.) Similar multiplications are required initially to account for any differences in the initial distribution.

**Application of Likelihood Ratios to Gradient Estimation**

Likelihood ratio techniques can be used to yield an estimate of \( f'(\theta) \) based on simulation at the single parameter value \( \theta \). Suppose that we have the representation \( f(\theta) = E_\theta[c(X, \theta)] \); we write \( E_\theta \) to indicate that the distribution of the random variable \( X \) depends on \( \theta \).

**Example:** Consider an M/M/1 queue with interarrival rate \( \lambda \) and service rate \( \theta \). A performance measure of above form is obtained by letting \( X \) be the average waiting time for the first \( k \) customers and setting \( c(x, \theta) = ax + bx \). This cost function reflects the tradeoff between operating costs due to an increase in the service rate and the costs due to increased waiting times for the customers. We assume that \( \theta > \lambda \), so that the queue is stable for all possible choices of \( \theta \).

As discussed above, we have

\[ f(\theta + h) = E_{\theta+h}[c(X, \theta + h)] = E_\theta[c(X, \theta + h)L(h)], \]

where \( L(h) \) is an appropriate likelihood ratio. Noting that \( L(0) = 1 \) and assuming that we can interchange the expectation and limit operations, we have

\[ f'(\theta) = \lim_{h \to 0} \frac{f(\theta + h) - f(\theta)}{h} = \lim_{h \to 0} E_\theta \left[ \frac{c(X, \theta + h)L(h) - c(X, \theta)L(0)}{h} \right]. \]
\[
E_{\theta}\left[\lim_{h \to 0} \frac{c(\mathbf{X}, \theta + h)L(h) - c(\mathbf{X}, \theta)L(0)}{h}\right] = E_{\theta}\left[\frac{d}{dh}\left(c(\mathbf{X}, \theta + h)L(h)\right)\right]_{h=0} = E_{\theta}\left[c'(\mathbf{X}, \theta) + c(\mathbf{X}, \theta)L'(0)\right]
\]

(Here \(c' = \partial c / \partial \theta\).) To estimate \(r(\theta) = f'(\theta)\), generate i.i.d. replicates \(X_1, \ldots, X_n\) and compute the estimator

\[
\hat{r}_n(\theta) = \hat{r}_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[ c'(X_i, \theta) + c(X_i, \theta)L'_i(0) \right],
\]

where \(L'_i(0)\) denotes the value of \(L'(0)\) that was computed during the \(i\)th simulation replication. Confidence intervals can be formed in the usual way. In the context of the Robbins-Monro algorithm (where computing confidence intervals is not an issue), we often take \(n\) to be small, even perhaps taking \(n = 1\), because in practice it is often more efficient to execute many R-M iterations with relatively noisy derivative estimates than to execute fewer iterations with more precise derivative estimates.

**Example:** For the M/M/1 queue discussed above, denote by \(V_1, \ldots, V_k\) the \(k\) service times generated in the course of simulating the \(k\) waiting times. Then the likelihood ratio \(L(h)\) is given by

\[
L(h) = \prod_{i=1}^{k} (\theta + h)e^{-(\theta + h)V_i} / \prod_{i=1}^{k} \theta e^{-\theta V_i} = \prod_{i=1}^{k} \left( \frac{\theta + h}{\theta} \right) e^{-\theta V_i},
\]

so that

\[
L'(0) = \sum_{i=1}^{k} \left( 1 - \frac{1}{\theta} - V_i \right) \quad \text{and} \quad \hat{r}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[ a(\theta + bX_i) \left( \sum_{j=1}^{k} \left( 1 - V_{ij} \right) \right) \right],
\]

where \(X_i\) is the average of the \(k\) waiting times generated during the \(j\)th simulation and \(V_{ij}\) is the \(i\)th service time generated during the \(j\)th simulation. Confidence intervals for \(f'(\theta)\) can be formed in the usual way.

**Remark:** When estimating gradients in the context of GSMP’s, the likelihood ratio often has the form \(L(h) = r_1(h) r_2(h) \cdots r_k(h)\) for some \(k \geq 1\), where each \(r_i(h)\) is a ratio of clock-setting density functions of the form \(f_0(x,s',e',s,E') / f_0(x,s',e',s,E')\) or perhaps a ratio of state-transition probabilities, so that \(r_i(0) = 1\) for each \(i\). Observe that, using the product rule for differentiation,
So, by induction, \( L'(0) = r_1(0) + \cdots + r_k(0) \). Note that if, for example, a factor \( r_i(h) \) is of the form \( f_{i+1}(X; s^*, e^*, s, E^*) / f_i(X; s^*, e^*, s, E^*) \), then

\[
\frac{d}{dh} \left. \frac{f_{i+1}(X; s^*, e^*, s, E^*)}{f_i(X; s^*, e^*, s, E^*)} \right|_{h=0} = r_i(0)
\]

Remark: Gradient information is useful even outside the context of stochastic optimization, because it can be used to explore the sensitivity of system performance to specified parameters.

Remark: There are other methods that can be used to estimate gradients, such as “infinitesimal perturbation analysis” (IPA). See Chapter 9 in the Handbook of Simulation for further discussion.

Remark: The main drawback to the likelihood-ratio approach is that the variance of the likelihood ratio increases with the length of the simulation. Thus the estimates can be very noisy for long simulation runs, and many simulation replications must be performed to get reasonable estimates.

### 2. Small Number of Alternatives: Ranking and Selection

Now let’s look at the case of a finite number of alternatives. Law and Kelton discuss the case of two competing alternatives in section 10.2. We focus on the more typical case where there are a number of alternatives to be compared. Unless stated otherwise, we will assume that “bigger is better,” i.e., our goal is to maximize some performance measure. (Any minimization problem can be converted to a maximization problem by maximizing the negative of the original performance measure.)

Suppose that we are comparing \( k \) systems, where \( k \) is on the order of 3 to 20. Denote by \( \mu_i \) the value of the performance measure for the \( i^{th} \) alternative. (E.g., \( \mu_i \) might be the expected revenue from system \( i \) over one year.) Sometimes we write \( \mu_{(1)} < \mu_{(2)} < \cdots < \mu_{(k)} \), so that \( \mu_{(k)} \) is the optimal value of the performance measure.

The goal is to select the best system with (high) probability \( 1 - \alpha \), whenever the performance measure for the best system is at least \( \delta \) better than the others; i.e., whenever \( \mu_{(k)} - \mu_{(i)} > \delta \) for \( i \neq k \). The parameter \( \delta \) is called the *indifference zone*. If in fact there exist solutions within the indifference zone (e.g., when \( \mu_{(k)} - \mu_{(i)} \leq \delta \) for one or more values of \( i \)), the algorithm described below typically selects a good solution (i.e., a system lying within the indifference zone) with probability \( \geq 1 - \alpha \).
The inputs to the procedure are observations $Y_{ij}$, where $Y_{ij}$ is the $j$th estimate of $\mu_i$. For a fixed alternative $i$, the observations $(Y_{ij}: j \geq 1)$ are assumed to be i.i.d. (i.e., based on independent simulation runs). Each $Y_{ij}$ is required to have (at least approximately) a normal distribution. Thus, an observation $Y_{ij}$ may be

- an average of a “large enough” number of i.i.d. observations of a finite-horizon performance measure so that the CLT is applicable
- an estimate of a steady-state performance measure based on a “large enough” number of regenerative cycles
- an estimate of a steady-state performance measure based on a “large enough” number of batch means

The following algorithm (due to Rinott, with modifications by Nelson and Matejcik) will not only select the “best” system as defined above, but will also provide simultaneous $100(1 - \alpha)$% confidence intervals on the quantities

$$\gamma_i = \mu_i - \max_{j \neq i} \mu_j$$

for $1 \leq i \leq k$. That is, $P(\gamma_i \in J_i \text{ for } 1 \leq k) = 1 - \alpha$. (This is a stronger assertion than $P(\gamma_i \in J_i) = 1 - \alpha$ for $1 \leq i \leq k$ as with individual confidence intervals.) This “multiple comparisons with the best” approach indicates how close each of the inferior systems is to being the best. This information is useful in identifying other good solutions; one of them may be needed if the “best” solution is ruled out because of secondary considerations (ease of implementation, maintenance costs, etc.). See *Handbook of Simulation*, Section 8.3.2 for further details.

**Algorithm**

1. Specify $\alpha$, $\delta$, and an initial sample size $n_0$ ($n_0 \geq 20$).
2. For $i = 1$ to $k$, take an i.i.d. sample $Y_{i1}, Y_{i2}, \ldots, Y_{in_0}$
3. For $i = 1$ to $k$, compute $\bar{Y}_i = \frac{1}{n_0} \sum_{j=1}^{n_0} Y_{ij}$ and $s_i^2 = \frac{1}{n_0 - 1} \sum_{j=1}^{n_0} (Y_{ij} - \bar{Y}_i)^2$
4. Look up the value of $h$ from Table 8.3 (given at the end of the lecture notes).
5. For $i = 1$ to $k$, take $N_i - n_0$ additional samples $Y_{ij}$ independent of the first-stage sample, where

$$N_i = \max \left( n_0, \left[ \frac{hs_i^2}{\delta} \right]^2 \right)$$

6. For $i = 1$ to $k$, compute $\alpha(i) = \frac{1}{N_i} \sum_{j=1}^{N_i} Y_{ij}$
7. Select the system with the largest value of $\alpha(i)$ as the best.

8. Form the simultaneous $100(1 - \alpha)$% confidence intervals for $\gamma_i$ as $[a_i, b_i]$, where

$$a_i = \min(0, \alpha(i) - \max_{j \neq i} \alpha(j) - \delta)$$

$$b_i = \max(0, \alpha(i) - \max_{j \neq i} \alpha(j) + \delta).$$

The justification of this algorithm is similar to (but more complicated than) the discussion in Section 10A in Law and Kelton. See Rinott’s 1978 paper in *Commun. Statist. A: Theory and Methods* for details. (The algorithm presented here is simpler to execute, however, than the algorithm discussed in Law and Kelton.)

**Remark:** Improved versions of this procedure can be obtained through the use of common random numbers. See *Handbook of Simulation*, Chapter 8.

### 3. Large Number of Alternatives: Discrete Stochastic Optimization

We wish to solve the problem $\max_{\theta \in \Theta} f(\theta)$, where the value of $f(\theta)$ is estimated using simulation and $\Theta$ is a very large set of discrete design alternatives. For example, $\theta$ might be the number of servers at a service center and $f(\theta)$ might be the expected value of the total revenue over a year of operation.

Development of good random search algorithms is a topic of ongoing research. Most of the current algorithms generate a sequence $(\theta_n : n \geq 0)$ with $\theta_{n+1} \in N(\theta_n) \cup \{\theta_n\}$, where $N(\theta)$ is a neighborhood of $\theta$. A currently-optimal solution $\theta_n^*$ is maintained. The algorithms differ in the structure of the neighborhood, the rule used to determine $\theta_{n+1}$ from $\theta_n$, the rule used to compute $\theta_n^*$, and the rule used to stop the algorithm.

We present a typical algorithm of this type (due to Andradottir) which uses a “simulated annealing”-type rule to determine $\theta_{n+1}$ from $\theta_n$. This rule usually accepts a better solution but, with a small probability, sometimes accepts a worse solution in order to avoid getting stuck at a local maximum.

**Algorithm**

1. Fix $T, N > 0$. Select a starting point $\theta_0$, and generate an estimate $\hat{f}(\theta_0)$ of $f(\theta_0)$ via simulation. Set $A(\theta_0) = \hat{f}(\theta_0)$ and $C(\theta_0) = 1$. Set $n = 0$ and $\theta_n^* = \theta_0$.

2. Select a candidate solution $\theta_n^*$ randomly and uniformly from $\Theta \setminus \{\theta_n\}$. Generate an estimate $\hat{f}(\theta_n^*)$ of $f(\theta_n^*)$ via simulation.
3. If \( \hat{f}(\theta_n^*) \geq \hat{f}(\theta_n) \), then set \( \theta_{n+1} = \theta_n^* \). Otherwise, generate a random variable \( U \) distributed uniformly on \([0, 1]\). If

\[
U \leq \exp \left[ \frac{\hat{f}(\theta_n^*) - \hat{f}(\theta_n)}{T} \right]
\]

then set \( \theta_{n+1} = \theta_n^* \); otherwise, set \( \theta_{n+1} = \theta_n \).

4. Set \( A(\theta_{n+1}) = A(\theta_{n+1}) + \hat{f}(\theta_{n+1}) \) and \( C(\theta_{n+1}) = C(\theta_{n+1}) + 1 \). If \( A(\theta_{n+1}) / C(\theta_{n+1}) > A(\theta^*_n) / C(\theta^*_n) \), then set \( \theta^*_{n+1} = \theta_{n+1} \); otherwise, set \( \theta^*_{n+1} = \theta^*_n \).

5. If \( n \geq N \) return the final answer \( \theta^*_{n+1} \) with estimated objective function value \( A(\theta^*_{n+1}) / C(\theta^*_{n+1}) \). Otherwise, set \( n = n + 1 \) and go to Step 2.

In the above algorithm, the neighborhood was defined as \( N(\theta) = \Theta - \{\theta\} \). If the problem has more structure, the neighborhood may be defined in a problem-specific manner.
<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\nu_0$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>5</td>
<td>2.291</td>
<td>3.058</td>
<td>3.511</td>
<td>3.837</td>
<td>4.093</td>
<td>4.305</td>
<td>4.486</td>
<td>4.644</td>
<td>4.786</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>2.107</td>
<td>2.758</td>
<td>3.126</td>
<td>3.384</td>
<td>3.582</td>
<td>3.744</td>
<td>3.881</td>
<td>3.999</td>
<td>4.103</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>1.948</td>
<td>2.510</td>
<td>2.817</td>
<td>3.027</td>
<td>3.186</td>
<td>3.314</td>
<td>3.420</td>
<td>3.512</td>
<td>3.592</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>1.906</td>
<td>2.447</td>
<td>2.739</td>
<td>2.938</td>
<td>3.088</td>
<td>3.208</td>
<td>3.308</td>
<td>3.393</td>
<td>3.467</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1.896</td>
<td>2.431</td>
<td>2.720</td>
<td>2.916</td>
<td>3.064</td>
<td>3.182</td>
<td>3.280</td>
<td>3.364</td>
<td>3.437</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>1.852</td>
<td>2.366</td>
<td>2.641</td>
<td>2.827</td>
<td>2.966</td>
<td>3.077</td>
<td>3.169</td>
<td>3.247</td>
<td>3.315</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>1.844</td>
<td>2.354</td>
<td>2.627</td>
<td>2.810</td>
<td>2.948</td>
<td>3.057</td>
<td>3.148</td>
<td>3.225</td>
<td>3.292</td>
</tr>
<tr>
<td>0.05</td>
<td>5</td>
<td>3.107</td>
<td>3.905</td>
<td>4.390</td>
<td>4.744</td>
<td>5.025</td>
<td>5.259</td>
<td>5.461</td>
<td>5.638</td>
<td>5.797</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>2.910</td>
<td>3.602</td>
<td>4.010</td>
<td>4.303</td>
<td>4.533</td>
<td>4.722</td>
<td>4.884</td>
<td>5.025</td>
<td>5.150</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>2.373</td>
<td>2.828</td>
<td>3.074</td>
<td>3.242</td>
<td>3.368</td>
<td>3.469</td>
<td>3.553</td>
<td>3.624</td>
<td>3.687</td>
</tr>
</tbody>
</table>