where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from $g(\mathbf{x})$. Note that $\widehat{\nabla^k \ell}(\mathbf{u})$ is an unbiased estimator of $\nabla^k \ell(\mathbf{u})$ for *all* $\mathbf{u}$. This means that, by varying $\mathbf{u}$ and keeping g fixed, we can, in principle, estimate unbiasedly the whole *response surface* $\{\nabla^k \ell(\mathbf{u}), \mathbf{u} \in \mathscr{V}\}$ from a *single simulation*. Often the importance sampling distribution is chosen in the *same* class of distributions as the original one. That is, $g(\mathbf{x}) = f(\mathbf{x}; \mathbf{v})$ for some $\mathbf{v} \in \mathscr{V}$. If not stated otherwise, we assume from now on that $g(\mathbf{x}) = f(\mathbf{x}; \mathbf{v})$; that is, we assume that the importance sampling pdf lies in the same parametric family as the original pdf $f(\mathbf{x}; \mathbf{u})$. If we denote the likelihood ratio estimator of $\ell(\mathbf{u})$ for a given $\mathbf{v}$ by $\widehat{\ell}(\mathbf{u}; \mathbf{v})$,

$$\widehat{\ell}(\mathbf{u}; \mathbf{v}) = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i)\, W(\mathbf{X}_i; \mathbf{u}, \mathbf{v}) \,, \tag{7.15}$$

with $W(\mathbf{x}; \mathbf{u}, \mathbf{v}) = f(\mathbf{x}; \mathbf{u})/f(\mathbf{x}; \mathbf{v})$, and likewise denote the estimators in (7.14) by $\widehat{\nabla^k \ell}(\mathbf{u}; \mathbf{v})$, then (see Problem 7.4)

$$\widehat{\nabla^k \ell}(\mathbf{u}; \mathbf{v}) = \nabla^k \widehat{\ell}(\mathbf{u}; \mathbf{v}) = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i)\, \mathcal{S}^{(k)}(\mathbf{u}; \mathbf{X}_i)\, W(\mathbf{X}_i; \mathbf{u}, \mathbf{v}) \,. \tag{7.16}$$

Thus the estimators of sensitivities are simply the sensitivities of the estimators.

Next, we apply importance sampling to the two toy examples 7.2 and 7.3, and show how to estimate $\nabla^k \ell(\mathbf{u})$ simultaneously for different values of $\mathbf{u}$ using a single simulation from the importance sampling pdf $f(\mathbf{x}; \mathbf{v})$.

■ **EXAMPLE 7.5 Example 7.2 (Continued)**

Consider again the Bernoulli toy example, with $H(\mathbf{X}) = X$ and $X \sim \mathsf{Ber}(u)$. Suppose that the importance sampling distribution is $\mathsf{Ber}(v)$, that is,

$$g(x) = f(x; v) = v^x (1 - v)^{1-x} \,, \quad x = 0, 1 \,.$$

Using importance sampling, we can write $\nabla^k \ell(u)$ as

$$\nabla^k \ell(u) = \mathbb{E}_v \left[X \frac{u^X (1-u)^{1-X}}{v^X (1-v)^{1-X}} \mathcal{S}^{(k)}(u; X) \right],$$

where $X \sim \mathsf{Ber}(v)$. Recall that for $\mathsf{Bin}(1, u)$ we have $\mathcal{S}(u; x) = \frac{x-u}{u(1-u)}$. The corresponding likelihood ratio estimator of $\nabla^k \ell(u)$ is

$$\begin{aligned} \widehat{\nabla^k \ell}(u; v) &= \frac{1}{N} \sum_{i=1}^{N} X_i \frac{u^{X_i} (1-u)^{1-X_i}}{v^{X_i} (1-v)^{1-X_i}} \mathcal{S}^{(k)}(u; X_i) \\ &= \frac{u}{v} \frac{1}{N} \sum_{i=1}^{N} X_i\, \mathcal{S}^{(k)}(u; X_i) \,, \end{aligned} \tag{7.17}$$

where $X_1, \ldots, X_N$ is a random sample from $\mathsf{Ber}(v)$. In the second equation we have used the fact that X_i is either 0 or 1. For $k = 0$ we readily obtain

$$\widehat{\ell}(u; v) = \frac{u}{v} \frac{1}{N} \sum_{i=1}^{N} X_i \,,$$

which also follows directly from (7.15), and for $k = 1$ we have

$$\widehat{\nabla \ell}(u; v) = \frac{u}{v} \frac{1}{N} \sum_{i=1}^{N} X_i \frac{X_i - u}{u(1-u)} = \frac{1}{v} \frac{1}{N} \sum_{i=1}^{N} X_i \,, \tag{7.18}$$

which is the derivative of $\widehat{\ell}(u;v)$, as observed in (7.16). Note that in the special case where $v = u$, the likelihood ratio estimators $\widehat{\ell}(u;u)$ and $\widehat{\nabla\ell}(u;u)$ reduce to the CMC estimator $\frac{1}{N}\sum_{i=1}^{N} X_i$ (sample mean) and the earlier derived score function estimator (7.10), respectively. As a simple illustration, we take a sample from $\mathsf{Ber}(v = 1/2)$ of size $N = 20$ and obtain

$$\{x_1, \ldots, x_{20}\} = \{0,1,0,0,1,0,0,1,1,1,0,1,0,1,1,0,1,0,1,1\} .$$

Suppose that, using this sample, we wish to estimate the quantities $\ell(u) = \mathbb{E}_u[X]$ and $\nabla\ell(u)$ *simultaneously* for $u = 1/4$ and $u = 1/10$. We readily obtain

$$\widehat{\ell}(u = 1/4; v = 1/2) = \frac{1/4}{1/2}\,\frac{11}{20} = \frac{11}{40} ,$$

$$\widehat{\ell}(u = 1/10; v = 1/2) = \frac{1/10}{1/2}\,\frac{11}{20} = \frac{11}{100} ,$$

and $\widehat{\nabla\ell}(u;v) = 11/10$ for both $u = 1/4$ and $1/10$.

■ **EXAMPLE 7.6 Example 7.3 (Continued)**

Let us consider the estimation of $\nabla^k\ell(u)$ simultaneously for several values of u in the second toy example, where $H(X) = X$ and $X \sim \mathsf{Exp}(u)$. Selecting the importance sampling distribution as

$$g(x) = f(x;v) = v\,\mathrm{e}^{-vx}, \quad x > 0$$

for some $v > 0$, and using (7.14), we can express $\nabla^k\ell(u)$ as

$$\nabla^k\ell(u) = \mathbb{E}_v\left[X\frac{u\,\mathrm{e}^{-uX}}{v\,\mathrm{e}^{-vX}}\,\mathcal{S}^{(k)}(u;X)\right] ,$$

where $X \sim \mathsf{Exp}(v)$ and (see Table 7.1) $\mathcal{S}(u;x) = \frac{1-ux}{u}$. The sample average estimator of $\nabla^k\ell(u)$ (see (7.14)) is

$$\widehat{\nabla^k\ell}(u;v) = \frac{1}{N}\sum_{i=1}^{N} X_i\frac{u\,\mathrm{e}^{-uX_i}}{v\,\mathrm{e}^{-vX_i}}\,\mathcal{S}^{(k)}(u;X_i) , \tag{7.19}$$

where $X_1, \ldots, X_N$ is a random sample from $\mathsf{Exp}(v)$. For $k = 0$, we have

$$\widehat{\ell}(u;v) = \frac{1}{N}\sum_{i=1}^{N} X_i\frac{u\,\mathrm{e}^{-uX_i}}{v\,\mathrm{e}^{-vX_i}} \approx \frac{1}{u} ,$$

and for $k = 1$ we obtain

$$\widehat{\nabla\ell}(u;v) = \frac{1}{N}\sum_{i=1}^{N} X_i\frac{u\,\mathrm{e}^{-uX_i}}{v\,\mathrm{e}^{-vX_i}}\frac{1-uX_i}{u} \approx -\frac{1}{u^2} , \tag{7.20}$$

which is the derivative of $\widehat{\ell}(u;v)$, as observed in (7.16). Note that in the particular case where $v = u$, the importance sampling estimators, $\widehat{\ell}(u;u)$ and $\widehat{\nabla\ell}(u;u)$, reduce to the sample mean (CMC estimator) and the SF estimator (7.11), respectively.

For a given importance sampling pdf $f(\mathbf{x};\mathbf{v})$, the algorithm for estimating the sensitivities $\nabla^k \ell(\mathbf{u})$, $k = 0, 1, \ldots$, for *multiple values* $\mathbf{u}$ *from a single simulation run* is given next.

Algorithm 7.2.1: Score Function Method with Importance Sampling

1 Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from the importance sampling pdf $f(\mathbf{x};\mathbf{v})$, which must be chosen in advance.

2 Calculate the sample performance $H(\mathbf{X}_i)$ and the scores $\mathcal{S}^{(k)}(\mathbf{u};\mathbf{X}_i)$, $i = 1, \ldots, N$, for the desired parameter value(s) $\mathbf{u}$.

3 Calculate $\widehat{\nabla^k \ell}(\mathbf{u};\mathbf{v})$ according to (7.16).

From Algorithm 7.2.1 it follows that in order to estimate the sensitivities $\nabla^k \ell(\mathbf{u})$, $k = 1, 2, \ldots$, all we need is to apply formula (7.16), which involves calculation of the performance $H(\mathbf{X}_i)$ and estimation of the scores $\mathcal{S}^{(k)}(\mathbf{u};\mathbf{X}_i)$ based on a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ obtained from the importance sampling pdf $f(\mathbf{x};\mathbf{v})$.

Confidence regions for $\nabla^k \ell(\mathbf{u})$ can be obtained by standard statistical techniques. In particular (e.g., see [19] and Section 1.11), $N^{1/2}\left[\widehat{\nabla^k \ell}(\mathbf{u};\mathbf{v}) - \nabla^k \ell(\mathbf{u})\right]$ converges to a multivariate normal random vector with mean zero and covariance matrix

$$\mathrm{Cov}_{\mathbf{v}}(H\,\mathcal{S}^{(k)}\,W) = \mathbb{E}_{\mathbf{v}}\left[H^2\,W^2\,\mathcal{S}^{(k)}\mathcal{S}^{(k)\,T}\right] - [\nabla^k \ell(\mathbf{u})][\nabla^k \ell(\mathbf{u})]^\top, \tag{7.21}$$

using the abbreviations $H = H(\mathbf{X})$, $\mathcal{S}^{(k)} = \mathcal{S}^{(k)}(\mathbf{u};\mathbf{x})$ and $W = W(\mathbf{X};\mathbf{u},\mathbf{v})$. From this point on, we will use these abbreviations when convenient, abbreviating $\mathcal{S}^{(1)}$ further to $\mathcal{S}$.

In particular, in the case $k = 0$, the variance of $\widehat{\ell}(\mathbf{u};\mathbf{v})$, under the importance sampling density $f(\mathbf{x};\mathbf{v})$, can be written as

$$\mathrm{Var}\left(\widehat{\ell}(\mathbf{u};\mathbf{v})\right) = \mathbb{E}_{\mathbf{v}}\left[H^2\,W^2\right] - \ell^2(\mathbf{u})\,. \tag{7.22}$$

The crucial issue clearly is how to choose a good importance sampling pdf that ensures low-variance estimates of $\ell(\mathbf{u})$ and $\nabla\ell(\mathbf{u})$. As we will see, this is not a simple task. We start with the variance of $\widehat{\ell}(\mathbf{u};\mathbf{v})$. We will show that for exponential families of the form (A.9) it can be derived explicitly. Specifically, with $\boldsymbol{\theta}$ taking the role of $\mathbf{u}$ and $\boldsymbol{\eta}$ the role of $\mathbf{v}$, we have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\eta}}\left[H^2(\mathbf{X})W^2(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta})\right] &= \mathbb{E}_{\boldsymbol{\theta}}\left[H^2(\mathbf{X})W(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta})\right] \\
&= \int H^2(\mathbf{x})\frac{c(\boldsymbol{\theta})}{c(\boldsymbol{\eta})}\,\mathrm{e}^{(\boldsymbol{\theta}-\boldsymbol{\eta})\cdot\mathbf{t}(\mathbf{x})}\,c(\boldsymbol{\theta})\,\mathrm{e}^{\boldsymbol{\theta}\cdot\mathbf{t}(\mathbf{x})}\,h(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
&= \frac{c^2(\boldsymbol{\theta})}{c(\boldsymbol{\eta})}\int H^2(\mathbf{x})\,\mathrm{e}^{(2\boldsymbol{\theta}-\boldsymbol{\eta})\cdot\mathbf{t}(\mathbf{x})}\,h(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
&= \frac{c^2(\boldsymbol{\theta})}{c(\boldsymbol{\eta})\,c(2\boldsymbol{\theta}-\boldsymbol{\eta})}\,\mathbb{E}_{2\boldsymbol{\theta}-\boldsymbol{\eta}}\left[H^2(\mathbf{X})\right] \\
&= \mathbb{E}_{\boldsymbol{\eta}}\left[W^2(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta})\right]\;\mathbb{E}_{2\boldsymbol{\theta}-\boldsymbol{\eta}}\left[H^2(\mathbf{X})\right]\,.
\end{aligned} \tag{7.23}$$

Note that $\mathbb{E}_{\boldsymbol{\eta}}\left[W^2(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta})\right] = \mathbb{E}_{\boldsymbol{\theta}}\left[W(\mathbf{X};\boldsymbol{\theta},\boldsymbol{\eta})\right]$.

Table 7.2 shows the expectations $\mathbb{E}_v\left[W^2(X;u,v)\right]$ for common exponential families in Tables A.1 and 7.1. Note that in Table 7.2 only *one* parameter is changed, the one denoted by u and changed to v. The values of $\mathbb{E}_v\left[W^2(X;u,v)\right]$ were calculated via (A.9) and (7.23). In doing so, we first reparameterized the distribution in terms of (A.9), with $\theta = \psi(u)$ and $\eta = \psi(v)$, and then calculated

$$\mathbb{E}_\eta\left[W^2(X;\theta,\eta)\right] = \frac{c^2(\theta)}{c(\eta)\,c(2\theta-\eta)}\,. \tag{7.24}$$

Last, we substituted u and v back in order to obtain the desired $\mathbb{E}_v\left[W^2(X;u,v)\right]$.

Table 7.2: $\mathbb{E}_v[W^2]$ for commonly used distributions.

Distribution	$f(x;u)$	$\theta=\psi(u)$	$c(\theta)$	$\mathbb{E}_v[W^2(X;u,v)]$
$\mathsf{Gamma}(\alpha,u)$	$\frac{u^\alpha x^{\alpha-1}\mathrm{e}^{-ux}}{\Gamma(\alpha)}$	$-u$	$\frac{(-\theta)^\alpha}{\Gamma(\alpha)}$	$\left(\frac{u^2}{v(2u-v)}\right)^\alpha$
$\mathsf{N}(u,\sigma^2)$	$\frac{1}{\sigma\sqrt{2\pi}}\,\mathrm{e}^{-\frac{1}{2}\left(\frac{x-u}{\sigma}\right)^2}$	$\frac{u}{\sigma^2}$	$\frac{\mathrm{e}^{-\frac{1}{2}\theta^2\sigma^2}}{\sigma\sqrt{2\pi}}$	$\mathrm{e}^{\left(\frac{u-v}{\sigma}\right)^2}$
$\mathsf{Weib}(\alpha,u)$	$\alpha u\,(ux)^{\alpha-1}\,\mathrm{e}^{-(ux)^\alpha}$	$-u^\alpha$	$\alpha\,\theta$	$\frac{(u/v)^{2\alpha}}{2\,(u/v)^\alpha-1}$
$\mathsf{Bin}(n,u)$	$\binom{n}{x}u^x(1-u)^{n-x}$	$\ln\left(\frac{u}{1-u}\right)$	$(1+\mathrm{e}^\theta)^{-n}$	$\left(\frac{u^2-2uv+v}{(1-u)v}\right)^n$
$\mathsf{Poi}(u)$	$\frac{u^x\mathrm{e}^{-u}}{x!}$	$\ln u$	$\mathrm{e}^{-\mathrm{e}^\theta}$	$\mathrm{e}^{\left(\frac{(u-v)^2}{v}\right)}$
$\mathsf{G}(u)$	$u(1-u)^{x-1}$	$\ln(1-u)$	$1-\mathrm{e}^\theta$	$\frac{u^2(v-1)}{v(u^2-2u+v)}$

Let us consider, first, the $\mathsf{Gamma}(\alpha,u)$ pdf. It readily follows that in order for the estimator $\widehat{\ell}(u;v)$ to be meaningful ($\mathrm{Var}(\widehat{\ell}(u;v)) < \infty$), one has to ensure that $2u - v > 0$, ($v < 2\,u$); otherwise, W will "blow up" the variance of the importance sampling estimator $\widehat{\ell}(u;v)$. A more careful analysis [19] (see also Proposition A.4.2 in the Appendix) indicates that in this case v should be chosen smaller than u (instead of smaller than $2u$) because the optimal importance sampling pdf $f(x;v^*)$ has a "fatter" tail than the original pdf $f(x;u)$. A similar result holds for the estimators of $\nabla^k\ell(\mathbf{u})$ and for other exponential families.

Let us consider, next, the multidimensional case $\mathbf{X} = (X_1,\ldots,X_n)$. Assume for concreteness that the $\{X_i\}$ are independent and $X_i \sim \mathsf{Exp}(u_i)$. It is not difficult to derive (see Problem 7.3) that in this case

$$\mathbb{E}_{\mathbf{v}}[W^2] = \mathbb{E}_{\mathbf{u}}[W] = \prod_{k=1}^{n}\frac{1}{1-\delta_k^2}\,, \tag{7.25}$$

where $\delta_k = (u_k - v_k)/u_k,\ k = 1,\ldots,n$ is the *relative perturbation* in u_k. For the special case where δ_k does not depend on k, say, $\delta_k = \delta$, $k = 1,\ldots,n$, we obtain

$$\mathrm{Var}_v(HW) = (1-\delta^2)^{-n}\,\mathbb{E}_{2u-v}\left[H^2\right] - \ell^2\,. \tag{7.26}$$

It should be noted that for fixed δ (even with $v < 2u$, which corresponds to $\delta < 1$), the variance of HW increases exponentially in n. For small values of δ, the first term on the right-hand side of (7.26) can be approximated by

$$(1-\delta^2)^{-n} = \exp\left\{-n\ln\left(1-\delta^2\right)\right\} \approx \exp\left\{n\delta^2\right\} \ ,$$

since for small x, $\ln(1+x) \approx x$. In fact, for the variance of HW to be manageably small, the value $n\delta^2$ should not be too large. That is, as n increases, δ^2 should satisfy

$$\delta^2 = \mathcal{O}(n^{-1}) \ . \tag{7.27}$$

As is shown in [19] an assumption similar to (7.27) applies for rather general distributions and, in particular, for the exponential family.

Formula (7.27) is associated with the so-called *trust region*, that is, the region where the likelihood ratio estimator $\widehat{\nabla^k \ell}(\mathbf{u};\mathbf{v})$ can be trusted to give a reasonably good approximation of $\nabla^k \ell(\mathbf{u})$. As an illustration, consider the case where $u_i = u$, $v_i = v$ for all i, and $n = 100$. In this case, the estimator $\widehat{\ell}(u;v)$ performs reasonably well for δ not exceeding 0.1, that is, when the relative perturbation in u is within 10%. For larger relative perturbations, the term $\mathbb{E}_v[W^2]$ "blows up" the variance of the estimators. Similar results also hold for the derivatives of $\ell(u)$.

The (negative) results concerning the unstable behavior of the likelihood ratio W and the rapid decrease of the trust region with the dimensionality n (see (7.27)) do not leave much room for importance sampling to be used for estimating $\nabla^k \ell(\mathbf{u})$, $k \geqslant 0$, in high dimensions. For such problems we therefore suggest the use of the score function estimators given in (7.9) (those that do not contain the likelihood ratio term W) as estimators of the true $\nabla^k \ell(\mathbf{u})$. For low-dimensional problems, say $n \leqslant 10$, the importance sampling estimator (7.14) for $\nabla^k \ell(\mathbf{u})$ could still be used, provided that the trust region is properly chosen, such as when the relative perturbation δ from the original parameter vector $\mathbf{u}$ does not exceed 10–20%. Even in this case, in order to prevent the degeneration of the importance sampling estimates, it is crucial to choose the reference parameter vector $\mathbf{v}$ such that the associated importance sampling pdf $f(\mathbf{x};\mathbf{v})$ has a "fatter" tail than the original pdf $f(\mathbf{x};\mathbf{u})$; see further Section A.4 of the Appendix.

7.3 SIMULATION-BASED OPTIMIZATION OF DESS

For the optimization program (P_0) in (7.4), let us suppose that the objective function

$$\ell_0(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_0(\mathbf{X};\mathbf{u}_2)]$$

and some of the constraint functions

$$\ell_j(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_j(\mathbf{X};\mathbf{u}_2)]$$

are not available in analytical form, so that in order to solve (P_0) we must resort to simulation-based optimization. This approach involves using the sample average versions, $\widehat{\ell}_0(\mathbf{u})$ and $\widehat{\ell}_j(\mathbf{u})$ instead of $\ell_0(\mathbf{u})$ and $\ell_j(\mathbf{u})$, respectively. Recall that the parameter vector $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ can have distributional and structural components.

Let us consider, first, a general treatment of simulation-based programs of type (P_0), with an emphasis on how to estimate the optimal solution $\mathbf{u}^*$ of the program

(P_0) using a *single simulation* run. Assume that we are given a random sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ from the pdf $f(\mathbf{x}; \mathbf{u}_1)$ and consider the following two cases.

Case A. Either of the following holds true:

1. It is too expensive to store long samples $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ and the associated sequences $\{\widehat{\ell}_j(\mathbf{u})\}$.

2. The sample performance, $\widehat{\ell}_j(\mathbf{u})$, cannot be computed simultaneously for different values of $\mathbf{u}$. However, we are allowed to set the control vector, $\mathbf{u}$, at any desired value $\mathbf{u}^{(t)}$ and then compute the random variables $\widehat{\ell}_j(\mathbf{u}^{(t)})$ and (quite often) the associated derivatives (gradients) $\widehat{\nabla \ell_j}(\mathbf{u})$, at $\mathbf{u} = \mathbf{u}^{(t)}$.

Case B. Both of the following hold true:

1. It is easy to compute and store the whole sample, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$.

2. Given a sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$, it is easy to compute the sample performance $\widehat{\ell}_j(\mathbf{u})$ for any desired value $\mathbf{u}$.

The main difference between Case *A* and Case *B* is that the former is associated with an *on-line* optimization approach, called *stochastic approximation*, while the latter is associated with an *off-line* optimization approach, called *stochastic counterpart optimization* or *sample average approximation*. For references on stochastic approximation and the stochastic counterpart method, we refer the reader to [11] and [19], respectively.

The following two subsections deal separately with the stochastic approximation and the stochastic counterpart methods.

7.3.1 Stochastic Approximation

Stochastic approximation originated with Robbins and Monro [14] and Kiefer and Wolfowitz [8]. Kiefer and Wolfowitz dealt with on-line minimization of *smooth convex* problems of the form

$$\min_{\mathbf{u}} \ell(\mathbf{u}), \quad \mathbf{u} \in \mathscr{V}, \tag{7.28}$$

and they assumed the feasible set $\mathscr{V}$ to be convex and that at any fixed-in-advance point $\mathbf{u} \in \mathscr{V}$ an estimate $\widehat{\nabla \ell}(\mathbf{u})$ of the true gradient $\nabla \ell(\mathbf{u})$ can be computed. Here we will apply their stochastic approximation in the context of simulation-based optimization.

The stochastic approximation method iterates in $\mathbf{u}$, by way of the following recursive formula:

$$\mathbf{u}^{(t+1)} = \Pi_{\mathscr{V}}(\mathbf{u}^{(t)} - \beta_t \, \widehat{\nabla \ell}(\mathbf{u}^{(t)})) \,, \tag{7.29}$$

where $\beta_1, \beta_2, \ldots$ is a sequence of positive step sizes and $\Pi_{\mathscr{V}}$ denotes the projection onto the set $\mathscr{V}$, that is, $\Pi_{\mathscr{V}}(\mathbf{u})$ is the point in $\mathscr{V}$ closest to $\mathbf{u}$. The projection $\Pi_{\mathscr{V}}$ is needed in order to enforce feasibility of the generated points $\{\mathbf{u}^{(t)}\}$. If the problem is unconstrained, that is, the feasible set $\mathscr{V}$ coincides with the whole space, then this projection is the identity mapping and can be omitted from (7.29).

It is readily seen that (7.29) represents a *gradient descent procedure* in which the exact gradients are replaced by their estimates. Indeed, if the exact value $\nabla\ell(\mathbf{u}^{(t)})$ of the gradient were available, then $-\nabla\ell(\mathbf{u}^{(t)})$ would give the direction of steepest descent at the point $\mathbf{u}^{(t)}$. This would guarantee that if $\nabla\ell(\mathbf{u}^{(t)}) \neq 0$, then moving along this direction the value of the objective function decreases, that is, $\ell(\mathbf{u}^{(t)} - \beta\,\nabla\ell(\mathbf{u}^{(t)})) < \ell(\mathbf{u}^{(t)})$ for $\beta > 0$ small enough. The iterative procedure (7.29) mimics this idea by using estimates of the gradients instead of the actual ones. Note further that a new random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ should be generated to calculate each $\widehat{\nabla\ell}(\mathbf{u}^{(t)})$, $t = 1, 2, \ldots$.

We will now present several alternative estimators $\widehat{\nabla\ell}(\mathbf{u})$ of $\nabla\ell(\mathbf{u})$ regarding the model in Example 7.1.

■ **EXAMPLE 7.7 Example 7.1 (Continued)**

As before, let $H(\mathbf{X}; u_3, u_4) = \max\{X_1 + u_3, X_2 + u_4\}$, where $\mathbf{X} = (X_1, X_2)$ is a two-dimensional vector with independent components, and $X_i \sim f_i(X; u_i)$, $i = 1, 2$. Here we are interested in estimating the four-dimensional vector $\nabla\ell(\mathbf{u})$, where $\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X}; \mathbf{u}_2)]$, $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) = (u_1, u_2, u_3, u_4)$, with respect to both the distributional parameter vector $\mathbf{u}_1 = (u_1, u_2)$, and the structural vector $\mathbf{u}_2 = (u_3, u_4)$.

We will consider three alternative estimators for $\nabla\ell(\mathbf{u})$. These estimators are called (a) *direct*, (b) *inverse-transform*, and (c) *push-out* estimators. More details on these estimators and their various applications are given in [17].

(a) *The direct estimator of* $\nabla\ell(\mathbf{u})$. We have

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X}; \mathbf{u}_2)], \tag{7.30}$$

$$\frac{\partial\ell(\mathbf{u})}{\partial u_1} = \mathbb{E}_{\mathbf{u}_1}\left[H(\mathbf{X}; \mathbf{u}_2)\nabla \ln f_1(X_1; u_1)\right] , \tag{7.31}$$

$$\frac{\partial\ell(\mathbf{u})}{\partial u_3} = \mathbb{E}_{\mathbf{u}_1}\left[\frac{\partial H(\mathbf{X}; \mathbf{u}_2)}{\partial u_3}\right] , \tag{7.32}$$

and similarly for $\partial\ell(\mathbf{u})/\partial u_2$ and $\partial\ell(\mathbf{u})/\partial u_4$. Here

$$\frac{\partial H(\mathbf{X}; \mathbf{u}_2)}{\partial u_3} = \begin{cases} 1, & \text{if } X_1 + u_3 > X_2 + u_4 \ , \\ 0, & \text{otherwise,} \end{cases} \tag{7.33}$$

and similarly for $\partial H(\mathbf{X}; \mathbf{u}_2)/\partial u_4$. The sample estimators of $\partial\ell(\mathbf{u})/\partial u_i$, $i = 1, \ldots, 4$, can be obtained *directly* from their expected-value counterparts — hence the name *direct estimators*. For example, the estimator of $\partial\ell(\mathbf{u})/\partial u_3$ can be written as

$$\widehat{\nabla\ell}_3^{(1)}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial H(\mathbf{X}_i; \mathbf{u}_2)}{\partial u_3} , \tag{7.34}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a sample from $f(\mathbf{x}; \mathbf{u}_1) = f_1(x_1; u_1)\, f_2(x_2; u_2)$, and similarly for the remaining estimators $\widehat{\nabla\ell}_i^{(1)}(\mathbf{u})$ of $\partial\ell(\mathbf{u})/\partial u_i$, $i = 1, 2, 4$.

(b) *The inverse-transform estimator of* $\nabla\ell(\mathbf{u})$. Using the inverse transformations $X_i = F_i^{-1}(Z_i; u_i)$, where $Z_i \sim \mathsf{U}(0, 1)$, $i = 1, 2$, we can write $H(\mathbf{X}; \mathbf{u}_2)$ alternatively as

$$\breve{H}(\mathbf{Z}; \mathbf{u}) = \max\{F_1^{-1}(Z_1; u_1) + u_3,\ F_2^{-1}(Z_2; u_2) + u_4\} ,$$

where $\mathbf{Z} = (Z_1, Z_1)$. The expected performance $\ell(\mathbf{u})$ and the gradient $\nabla\ell(\mathbf{u})$ can be now written as

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathsf{U}}[\breve{H}(\mathbf{Z};\mathbf{u})]$$

and

$$\nabla\ell(\mathbf{u}) = \mathbb{E}_{\mathsf{U}}[\nabla\breve{H}(\mathbf{Z};\mathbf{u})] \,,$$

respectively. Here U denotes the uniform distribution. It is readily seen that in the inverse-transform setting all four parameters u_1, u_2, u_3, u_4 become *structural* parameters. The estimator of $\nabla\ell(\mathbf{u})$ based on the inverse-transform method, denoted as $\widehat{\nabla\ell}^{(2)}(\mathbf{u})$, is therefore

$$\widehat{\nabla\ell}^{(2)}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} \nabla\breve{H}(\mathbf{Z}_i;\mathbf{u}) \,, \tag{7.35}$$

where the partial derivatives of $\breve{H}(\mathbf{z};\mathbf{u})$ can be obtained similarly to (7.33). Note that the first-order derivatives of $\breve{H}(\mathbf{z};\mathbf{u})$ are piecewise-continuous functions with discontinuities at points for which $F_1^{-1}(z_2;u_1)+u_2 = F_2^{-1}(z_2;u_2)+u_4$.

(c) *The push-out estimator of* $\nabla\ell(\mathbf{u})$. Define the following two random variables: $\widetilde{X}_1 = X_1 + u_3$ and $\widetilde{X}_2 = X_2 + u_4$. Now, the original sample performance $H(\mathbf{X};u_3,u_4) = \max\{X_1+u_3, X_2+u_4\}$ and the expected value $\ell(\mathbf{u})$ can be written as $\widetilde{H}(\widetilde{\mathbf{X}}) = \max\{\widetilde{X}_1,\widetilde{X}_2\}$ and as

$$\ell(\mathbf{u}) = \mathbb{E}_{\widetilde{f}}[\widetilde{H}(\widetilde{\mathbf{X}})] = \mathbb{E}_{\widetilde{f}}[\max\{\widetilde{X}_1,\widetilde{X}_2\}] \,, \tag{7.36}$$

respectively. Here $\widetilde{f}$ is the pdf of $\widetilde{\mathbf{X}}$; thus, $\tilde{f}(\mathbf{x};\mathbf{u}) = f_1(x_1-u_3;u_1)\, f_2(x_2-u_4;u_2) = \tilde{f}_1(x;u_1,u_3)\,\tilde{f}_2(x;u_2,u_4)$. In this case we say that the original structural parameters u_3 and u_4 in $H(\cdot)$ are "pushed out" into the pdf $\tilde{f}$.

As an example, suppose that $X_j \sim \mathsf{Exp}(u_j),\ j=1,2$. Then the cdf $\widetilde{F}_1(x)$ of $\widetilde{X}_1 = X_1+u_3$ and the cdf $\widetilde{F}_2(x)$ of $\widetilde{X}_2 = X_2+u_4$ can be written, respectively, as

$$\widetilde{F}_1(x) = P(\widetilde{X}_1 \leqslant x) = \mathbb{P}(X_1 \leqslant x-u_3) = F_1(x-u_3)$$

and

$$\widetilde{F}_2(x) = \mathbb{P}(\widetilde{X}_2 \leqslant x-u_4) = F_2(x-u_4) \,.$$

Clearly,

$$\widetilde{f}_1(x;u_1,u_3) = \begin{cases} u_1\,\mathrm{e}^{-u_1(x-u_3)}, & x \geqslant u_3 \,, \\ 0 & \text{otherwise,} \end{cases} \tag{7.37}$$

and

$$\widetilde{f}_2(x;u_2,u_4) = \begin{cases} u_2\,\mathrm{e}^{-u_2(x-u_4)}, & x \geqslant u_4 \\ 0 & \text{otherwise.} \end{cases} \tag{7.38}$$

Because in representation (7.36) all four parameters $u_1,\ldots,u_4$ are *distributional*, in order to estimate $\nabla^k\ell(\mathbf{u})$, we can apply the SF method. In particular, we can estimate the gradient

$$\nabla\ell(\mathbf{u}) = \mathbb{E}_{\widetilde{f}}[\max\{\widetilde{X}_1,\widetilde{X}_2\}\,\nabla\ln\widetilde{f}(\widetilde{\mathbf{X}};\mathbf{u})]$$

as

$$\widehat{\nabla\ell}^{(3)}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} \max\{\widetilde{X}_{1i}, \widetilde{X}_{2i}\}\nabla \ln \widetilde{f}(\widetilde{\mathbf{X}}_i; \mathbf{u}) \ . \tag{7.39}$$

Recall that $\partial H(\mathbf{X};\mathbf{u}_1)/\partial u_1$ and $\partial H(\mathbf{X};u_1)/\partial u_2$ are piecewise-constant functions (see (7.33)) and that $\partial^2 H(\mathbf{X};\mathbf{u}_1)/\partial u_1^2$ and $\partial^2 H(\mathbf{X};\mathbf{u}_1)/\partial u_2^2$ vanish almost everywhere. Consequently, the associated second-order derivatives cannot be interchanged with the expectation operator in (7.30). Yet, the transformed function $\ell(\mathbf{u})$ in (7.36) and its sample version $\widehat{\nabla\ell}^{(3)}(\mathbf{u})$ are both *differentiable* in $\mathbf{u}$ everywhere, provided that $f_1(x_1 - u_3; u_1)$ and $f_2(x_2 - u_4; u_2)$ are smooth. Thus, subject to smoothness of $\tilde{f}(\mathbf{x};\mathbf{u})$, the push-out technique *smoothes out* the original *non-smooth* performance measure $H(\cdot)$.

Let us return to stochastic optimization. As we will see in Theorem 7.3.1 below, starting from some fixed initial value $\mathbf{u}^{(1)}$, and under some reasonable assumptions, the sequence $\{\mathbf{u}^{(t)}\}$ converges asymptotically in t to the minimizer $\mathbf{u}^*$ of the objective function $\ell(\mathbf{u})$ over $\mathscr{V}$. Typically, in order to guarantee the convergence, the following two conditions are imposed on the step sizes: (a) $\sum_{t=1}^{\infty}\beta_t = \infty$, and (b) $\sum_{t=1}^{\infty}\beta_t^2 < \infty$. For example, take $\beta_t \equiv c/t$ for some $c > 0$. There are many theorems on the convergence and rate of convergence for stochastic optimization. One of the simplest, from [11], is presented next.

Theorem 7.3.1 *Assume that ℓ is smooth and strictly convex, that is,*

$$\ell(\mathbf{u}+\mathbf{h}) \geqslant \ell(\mathbf{u}) + [\nabla\ell(\mathbf{u})]^\top \mathbf{h} + \frac{\beta}{2}\mathbf{h}^\top\mathbf{h}, \quad \beta > 0 \ . \tag{7.40}$$

Assume further that the errors in the stochastic gradient vector $\widehat{\nabla\ell}(\mathbf{u})$ have a bounded second moment, that is,

$$\mathbb{E}\left[\|\widehat{\nabla\ell}(\mathbf{u}) - \nabla\ell(\mathbf{u})\|^2\right] \leqslant C^2 < \infty \ .$$

Then, for an arbitrary (deterministic) positive sequence $\{\beta_t\}$ such that

$$\sum_{t=1}^{\infty}\beta_t = \infty, \quad \sum_{t=1}^{\infty}\beta_t^2 < \infty \ ,$$

the vector $\mathbf{u}^{(t)}$ converges asymptotically to $\mathbf{u}^*$ (the minimizer of $\ell(\mathbf{u})$) in the sense of mean square. If, moreover,

$$\beta_t = c/t \ ,$$

with an appropriate constant c (whether a given c is appropriate depends only on β in (7.40)), then for all t we have the following bounds:

$$\mathbb{E}\left[\|\mathbf{u}^{(t)} - \mathbf{u}^*\|^2\right] \leqslant \frac{A(\beta,c)}{t}\,\|\mathbf{u}^{(1)} - \mathbf{u}^*\|^2$$

and

$$\mathbb{E}\left[\ell(\mathbf{u}^{(t)}) - \ell(\mathbf{u}^*)\right] \leqslant \mathcal{O}(1/t) \ ,$$

where $A(\beta,c)$ is some constant depending on β and c.

The attractive features of the stochastic approximation method are its simplicity and ease of implementation in those cases in which the projection $\Pi_{\mathscr{Y}}(\cdot)$ can be easily computed. However, it also has severe shortcomings. The crucial question in its implementation is the choice of the step sizes $\{\beta_t\}$. Small step sizes slow the progress toward the optimum, and large step sizes cause the iterations to "zigzag". Also a few wrong steps in the beginning of the procedure may require many iterations to correct. For instance, the algorithm is extremely sensitive to the choice of the constant c in the step size rule $\beta_t = c/t$. Therefore, various step size rules have been suggested in which the step sizes are chosen adaptively. An easy way to balance the need for both small and large step sizes, is to use a sequence $\{\beta_t\}$ of the form

$$\beta_t = \frac{a}{(t+1+A)^\alpha},$$

for $A \geqslant 0$ and $0 \leqslant a \leqslant 1$, as suggested in Spall [21, Section 4.4]. Another drawback of the stochastic approximation method is that it lacks good stopping criteria and often has difficulties handling even relatively simple linear constraints.

Last, we would like to stress that even in cases where the direct, push-out, or inverse-transform estimators are not applicable (e.g., when H or f are not known or are difficult to evaluate), it is still possible to obtain estimates of the gradient of $\ell(\mathbf{u})$ via simulation, by using difference estimators. In particular, the *central difference estimator* of the i-th component of $\nabla\ell(\mathbf{u})$, that is, $\partial\ell(\mathbf{u})/\partial u_i$, is given by

$$\frac{\widehat{\ell}(\mathbf{u}+\mathbf{e}_i\,\delta/2) - \widehat{\ell}(\mathbf{u}-\mathbf{e}_i\,\delta/2)}{\delta},$$

where $\mathbf{e}_i$ denotes the i-th unit vector, and $\widehat{\ell}(\mathbf{u}+\mathbf{e}_i\,\delta/2)$ and $\widehat{\ell}(\mathbf{u}-\mathbf{e}_i\,\delta/2)$ can be any estimators of $\ell(\mathbf{u}+\mathbf{e}_i\,\delta/2)$ and $\ell(\mathbf{u}-\mathbf{e}_i\,\delta/2)$, respectively. The difference parameter δ should be small enough to reduce the bias of the estimator (which is of order $\mathcal{O}(\delta^2)$, see [2, Page 209]), but large enough to keep the variance of the estimator small.

It is important to use *common random variables* (CRVs) in the implementation, as this reduces the variance of the difference estimator; see Section 5.2. For the case where $\ell(\mathbf{u}) = \mathbb{E}[h(\mathbf{U};\mathbf{u})]$, with $\mathbf{U} \sim \mathsf{U}(0,1)^d$, for some function h, this leads to the following algorithm:

Algorithm 7.3.1: Central Difference Estimation with CRVs

1 Generate $\mathbf{U}_1,\ldots,\mathbf{U}_N \sim \mathsf{U}(0,1)^d$.

2 Let $L_k \leftarrow h(\mathbf{U}_k;\mathbf{u}-\mathbf{e}_i\,\delta/2)$ and $R_k \leftarrow h(\mathbf{U}_k;\mathbf{u}+\mathbf{e}_i\,\delta/2)$, $k=1,\ldots,N$.

3 Compute the sample covariance matrix corresponding to the pairs $\{(L_k,R_k)\}$:

$$C = \begin{pmatrix} \frac{1}{N-1}\sum_{k=1}^N (L_k-\bar{L})^2 & \frac{1}{N-1}\sum_{k=1}^N (L_k-\bar{L})(R_k-\bar{R}) \\ \frac{1}{N-1}\sum_{k=1}^N (L_k-\bar{L})(R_k-\bar{R}) & \frac{1}{N-1}\sum_{k=1}^N (R_k-\bar{R})^2 \end{pmatrix}.$$

4 Estimate $\partial\ell(\mathbf{u})/\partial\theta_i$ via the central difference estimator $\dfrac{\bar{R}-\bar{L}}{\delta}$, with an estimated variance of

$$\mathrm{SE} = \frac{C_{1,1} + C_{2,2} - 2\,C_{1,2}}{\delta^2\,N}.$$

7.3.2 Stochastic Counterpart Method

The underlying idea in the stochastic counterpart approach is to replace all the expected value functions in the deterministic program (P_0) by their sample average equivalents and then solve the latter by standard mathematical programming techniques. The resultant optimal solution provides an estimator of the corresponding true optimal solution of the original program (P_0).

If not stated otherwise, we will consider here the unconstrained program

$$\min_{\mathbf{u}\in\mathscr{V}} \ell(\mathbf{u}) = \min_{\mathbf{u}\in\mathscr{V}} \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X};\mathbf{u}_2)] \,. \tag{7.41}$$

The general constrained program (P_0) is treated in [19].

Assume that $\mathscr{V}$ is an open set and that $\ell(\mathbf{u})$ is continuously differentiable on $\mathscr{V}$. Then, by the first-order necessary conditions, the gradient of $\ell(\mathbf{u})$ at an optimal solution point, $\mathbf{u}^*$, must vanish. Consequently, the optimal solution $\mathbf{u}^*$ can be found by solving the equation system

$$\nabla\ell(\mathbf{u}) = \mathbf{0} \,, \quad \mathbf{u}\in\mathscr{V} \,. \tag{7.42}$$

Using the importance sampling pdf $f(\mathbf{x};\mathbf{v}_1)$, we can write the stochastic counterpart of (7.41) as

$$\min_{\mathbf{u}\in\mathscr{V}} \widehat{\ell}(\mathbf{u};\mathbf{v}_1) = \min_{\mathbf{u}\in\mathscr{V}} \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i;\mathbf{u}_2)\, W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1) \,, \tag{7.43}$$

where $\mathbf{X}_1,\ldots,\mathbf{X}_N$ is a random sample from the importance sampling pdf $f(\mathbf{x};\mathbf{v}_1)$ and

$$W(\mathbf{x};\mathbf{u}_1,\mathbf{v}_1) = \frac{f(\mathbf{x};\mathbf{u}_1)}{f(\mathbf{x};\mathbf{v}_1)} \,.$$

Assuming further that $\widehat{\ell}(\mathbf{u};\mathbf{v}_1)$ is continuously differentiable on $\mathscr{V}$, we find the optimal solution of (7.43) by solving the equation system

$$\widehat{\nabla\ell}(\mathbf{u};\mathbf{v}_1) = \nabla\widehat{\ell}(\mathbf{u};\mathbf{v}_1) = \mathbf{0} \,, \quad \mathbf{u}\in\mathscr{V} \,, \tag{7.44}$$

which itself may be viewed as a stochastic counterpart of the deterministic system (7.42). Thus we simply take the gradient of the likelihood ratio estimator $\widehat{\ell}(\mathbf{u};\mathbf{v}_1)$ as an estimator for the gradient of ℓ at $\mathbf{u}$; see also (7.16).

Note that (7.44) can be written as

$$\begin{aligned}\nabla\widehat{\ell}(\mathbf{u};\mathbf{v}_1) = \frac{1}{N}\sum_{i=1}^{N}\{&H(\mathbf{X}_i;\mathbf{u}_2)\nabla\ln f(\mathbf{X}_i;\mathbf{u}_1)\, W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1)\\ &+\nabla H(\mathbf{X}_i;\mathbf{u}_2)\, W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1)\} = \mathbf{0}, \quad \mathbf{u}\in\mathscr{V}.\end{aligned} \tag{7.45}$$

Recall that we can view the above problem as the sample average approximation of the true (or expected value) problem (7.41). The function $\widehat{\ell}(\mathbf{u};\mathbf{v}_1)$ is random in the sense that it depends on the corresponding sample $\mathbf{X}_1,\ldots,\mathbf{X}_N$. However, note that once the sample is generated, $\widehat{\ell}(\mathbf{u};\mathbf{v}_1)$ becomes a deterministic function whose values and derivatives can be computed for a given value of the argument $\mathbf{u}$. Consequently, the problem (7.43) becomes a deterministic optimization problem,

and one can solve it with an appropriate deterministic optimization algorithm. For example, in the unconstrained case it can be solved by using, say, the steepest descent method, that is,

$$\mathbf{u}^{(t+1)} = \Pi_{\mathscr{V}}(\mathbf{u}^{(t)} - \beta_t \nabla \widehat{\ell}(\mathbf{u}^{(t)}; \mathbf{v}_1)) \;, \tag{7.46}$$

where the step size β_t is obtained by a line search, for example,

$$\beta_t \equiv \underset{\beta}{\operatorname{argmin}} \{\widehat{\ell}(\mathbf{u}^{(t)} - \beta \nabla \widehat{\ell}(\mathbf{u}^{(t)}; \mathbf{v}_1); \mathbf{v}_1)\} \;,$$

and, as before, $\Pi_{\mathscr{V}}$ denotes the projection onto the set $\mathscr{V}$. Note that this procedure is different from the stochastic approximation method (7.29) in three respects:

1. The step sizes β_t are calculated by a line search instead of being defined a priori.
2. The same sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is used for all $\nabla \widehat{\ell}(\mathbf{u}^{(t)}; \mathbf{v}_1)$.
3. Typically, a reasonably large sample size N is used in (7.46) as compared to stochastic optimization in (7.29).

Next, we consider the particular case of the program (7.41) where $\mathbf{u}$ is a distributional decision parameter vector, that is, we consider the following program:

$$\min_{\mathbf{u} \in \mathscr{V}} \ell(\mathbf{u}) = \min_{\mathbf{u} \in \mathscr{V}} \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})] \;. \tag{7.47}$$

To estimate the optimal solution $\mathbf{u}^*$ of the program (7.47), we will use the score function method. In this case the program (7.43) reduces to

$$\min_{\mathbf{u} \in \mathscr{V}} \widehat{\ell}(\mathbf{u}; \mathbf{v}) = \min_{\mathbf{u} \in \mathscr{V}} \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i)\, W(\mathbf{X}_i; \mathbf{u}, \mathbf{v}) \;, \tag{7.48}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from the importance sampling pdf $f(\mathbf{x}; \mathbf{v})$, and

$$W(\mathbf{x}; \mathbf{u}, \mathbf{v}) = \frac{f(\mathbf{x}; \mathbf{u})}{f(\mathbf{x}; \mathbf{v})} \;.$$

By analogy to (7.45), we have

$$\frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i)\, \nabla \ln f(\mathbf{X}_i; \mathbf{u})\, W(\mathbf{X}_i; \mathbf{u}, \mathbf{v}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{V}. \tag{7.49}$$

Remark 7.3.1 The size of the trust region is an important consideration when solving the stochastic counterpart (7.48) based on likelihood ratios. In particular, the following two requirements must hold:

(a) The reference parameter vector $\mathbf{v}$ must be chosen carefully so that that the importance sampling pdf $f(\mathbf{x}; \mathbf{v})$ has a "fatter" tail than the original pdf $f(\mathbf{x}; \mathbf{u})$ (see also Section A.4 of the appendix); otherwise, the likelihood ratio $W(\mathbf{x}; \mathbf{u}, \mathbf{v})$ is likely to degenerate.

(b) The trust region $\mathscr{V}$ of all possible values of $\mathbf{v}$ should be decided in advance so that $\mathscr{V}$ is not too wide. In particular, it should satisfy (7.27). If the region $\mathscr{V}$

is too wide, the likelihood ratio term $W(\mathbf{x};\mathbf{u},\mathbf{v})$ will "blow up" the variance of the estimate of (7.48). In this occurs, alternative methods (not involving likelihood ratio terms), like the steepest descent method (7.46), should be used. Common alternatives are the steepest descent and the stochastic approximation methods.

EXAMPLE 7.8 Stochastic Shortest Path

Consider again the stochastic shortest path problem in Example 5.15 on Page 158. Let the components $\{X_i\}$ be independent and $X_i \sim \mathsf{Exp}(u_i^{-1})$, $i = 1,\ldots,5$, with $\mathbf{u} = (1,2,u,4,5)$, for some $u > 0$. Our objective now is to solve the following program:

$$\min_{u\in\mathscr{V}} \ell(u) = \min_{u\in\mathscr{V}} [3.1 - \mathbb{E}_u[S(\mathbf{X})] + 0.1\,u] \,, \tag{7.50}$$

where $\mathscr{V} = \{u : 1 \leqslant u \leqslant 4\}$, $S(\mathbf{X}) = \min\{X_1 + X_4,\ X_1 + X_3 + X_5,\ X_2 + X_3 + X_4,\ X_2 + X_5\}$ denotes the length of the shortest path. The function $\ell(u)$ is difficult to evaluate exactly but can be estimated easily via Monte Carlo simulation, yielding a "noisy" function $\widehat{\ell}(u)$. Figure 7.1 displays estimates and confidence intervals for various values of u, using for each estimate a sample size of $N = 50{,}000$. The minimizer, say u^*, seems to lie in the interval [1,2].

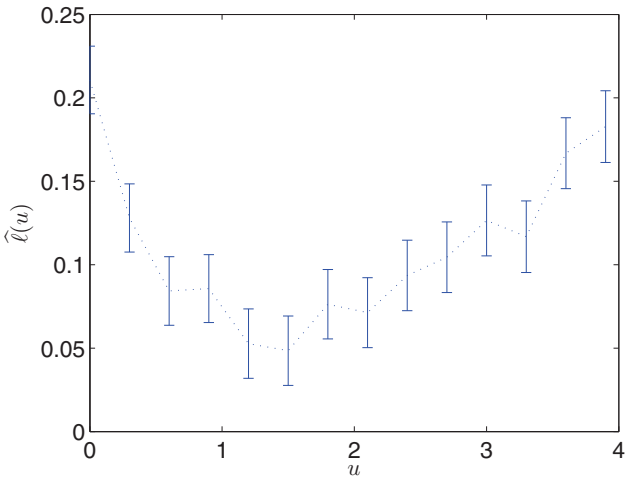

Figure 7.1: Minimize $\ell(u)$ with respect to u. Estimates and 95% confidence intervals indicate that the minimum is attained between 1 and 2.

To find u^* via the stochastic counterpart method, we can proceed as follows. First, the derivative of $\ell(u)$ satisfies

$$\begin{aligned}\nabla\ell(u) &= \frac{1}{10} - \mathbb{E}_v[S(\mathbf{X})\,\mathcal{S}(u;\mathbf{X})\,W(\mathbf{x};u,v)] \\ &= \frac{1}{10} - \mathbb{E}_v\left[S(\mathbf{X})\frac{X_3-u}{u^2}\frac{v}{u}\,\mathrm{e}^{-X_3(u^{-1}-v^{-1})}\right] \,,\end{aligned}$$

where the score function $\mathcal{S}(u;\mathbf{X})$ is given by

$$\mathcal{S}(u;\mathbf{X}) = \frac{\partial}{\partial u} \ln\left(u^{-1}\mathrm{e}^{-X_3/u}\right) = \frac{X_3 - u}{u^2} \,. \tag{7.51}$$

Hence, the stochastic counterpart of $\nabla\ell(u) = 0$ is

$$\widehat{\nabla\ell}(u;v) = \frac{1}{10} - \frac{v}{N}\sum_{i=1}^{N} S(\mathbf{X}_i)\frac{X_{3i}-u}{u^3}\,\mathrm{e}^{-X_{3i}(u^{-1}-v^{-1})} = 0 \,,$$

with $\mathbf{X}_1,\ldots,\mathbf{X}_N$ simulated under parameter v. The choice of v here is *crucial*. In [17] the following method is proposed for choosing a "good" parameter v. We start by imposing some constraints on v, that is, $v_1 \leqslant v \leqslant v_2$, reflecting our prior knowledge about the optimal u^* (lying between v_1 and v_2). In our case, we could take $v_1 = 1$ and $v_2 = 4$, for example, since $\mathscr{V} = \{u : \leqslant u \leqslant 4\}$. Next, we take v from this interval such that the tail of the corresponding distribution is as fat as possible to ensure that the likelihood ratio behaves well. In this case, it means taking $v = 4$.

Figure 7.2 shows the graph of $\widehat{\nabla\ell}(u;v)$ as a function of u (solid line). It was obtained from a sample of size $N = 500{,}000$ using $v = 4$. Recall that, by definition, $\nabla\ell(u^*) = 0$. As an estimate for u^* we take the point $\widehat{u^*}$ such that $\widehat{\nabla\ell}(\widehat{u^*};v) = 0$. This can be found by standard root-finding procedures, such as Matlab's `fzero` function. In this example, we find that $\widehat{u^*} = 1.37$.

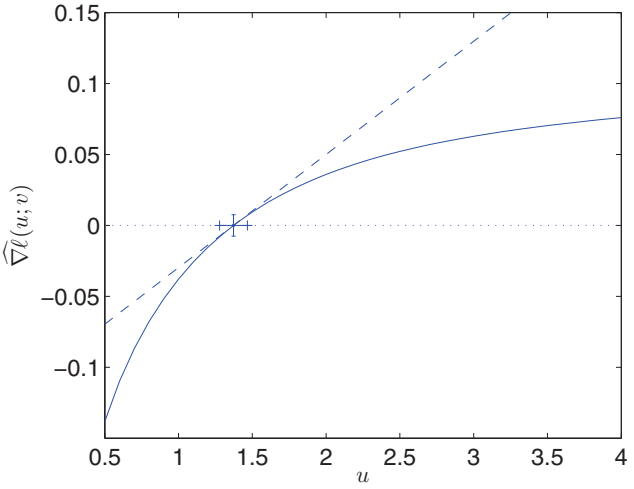

Figure 7.2: Estimate and 95% confidence interval for u^*. The estimate is found as the root of $\widehat{\nabla\ell}(u;v)$.

We can proceed to describe how to construct a confidence interval for u^* using a confidence interval for $\nabla\ell(u^*)$. For ease of notation, we write $g(u)$ for $\nabla\ell(u)$ and $\widehat{g}(u)$ for its estimate, $\widehat{\nabla\ell}(u;v)$, and we assume that both $g(u)$ and

$\widehat{g}(u)$ are monotonically increasing. Since $\widehat{g}(u)$ is of the form

$$\widehat{g}(u) = \frac{1}{10} - \frac{1}{N}\sum_{i=1}^{N} Z_i \,,$$

where $\{Z_i\}$ are iid random variables, an approximate $(1-\alpha)$ confidence interval for $g(u^*) = 0$ is $(-C, C)$, with $C = z_{1-\alpha/2}\, S_Z/\sqrt{N}$. Here S_Z is the sample standard deviation of the $\{Z_i\}$ and $z_{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the standard normal distribution. As a consequence, $(g^{-1}(-C), g^{-1}(C))$ is an approximate $(1-\alpha)$ confidence interval for u^*. For small C we have $g^{-1}(C) \approx u^* + C/g'(u^*)$, where g' is the derivative of g, that is, the second derivative of ℓ. The latter is given by

$$g'(u) = \nabla^2 \ell(u) = -\mathbb{E}_v \left[S(\mathbf{X})\, \frac{2u^2 - 4uX_3 + X_3^2}{u^4}\, \frac{v}{u}\, \mathrm{e}^{-X_3(u^{-1}-v^{-1})} \right]$$

and it can be estimated via its stochastic counterpart, using the same sample as we used to obtain $\widehat{g}(u)$. Indeed, the estimate of $g'(u)$ is simply the derivative of $\widehat{g}$ at u. Thus, an approximate $(1-\alpha)$ confidence interval for u^* is $\widehat{u^*} \pm C/\widehat{g}'(\widehat{u^*})$. This is illustrated in Figure 7.2, where the dashed line corresponds to the tangent line to $\widehat{g}(u)$ at the point $(\widehat{u}, 0)$, and 95% confidence intervals for $g(\widehat{u})$ and u^* are plotted vertically and horizontally, respectively. The particular values for these confidence intervals were found to be $(-0.0075, 0.0075)$ and $(1.28, 1.46)$.

Last, it is important to choose the parameter v under which the simulation is carried out greater than u^*. This is shown in Figure 7.3, where 10 replications of $\widehat{g}(u)$ are plotted for cases $v = 0.5$ and $v = 4$.

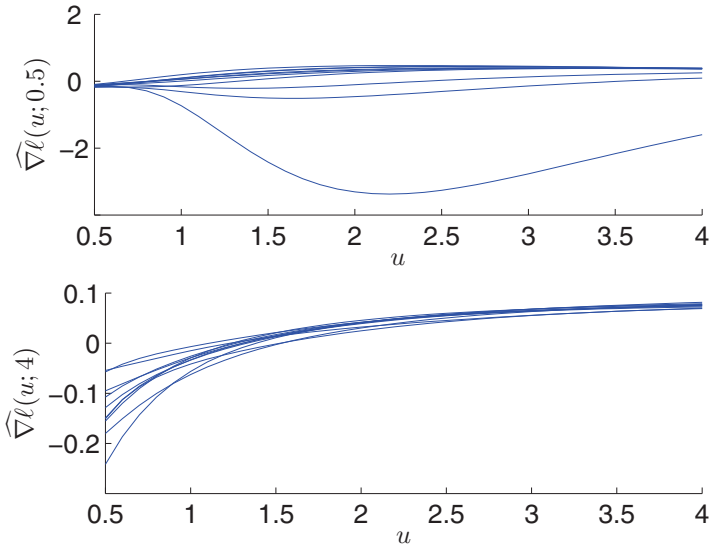

Figure 7.3: Ten replications of $\widehat{\nabla}\ell(u;v)$ are simulated under $v = 0.5$ and $v = 4$.

Note that in the first case the estimates of $g(u) = \widehat{\nabla \ell}(u; v)$ fluctuate widely, whereas in the second case they remain stable. As a consequence, u^* cannot be reliably estimated under $v = 0.5$. For $v = 4$ no such problems occur. All this is in accordance with the general principle that the importance sampling distribution should have heavier tails than the target distribution. Specifically, under $v = 4$ the pdf of X_3 has heavier tails than under $v = u^*$, whereas the opposite is true for $v = 0.5$.

In general, let $\widehat{\ell}^*$ and $\widehat{\mathbf{u}}^*$ denote the optimal objective value and the optimal solution of the sample average problem (7.48), respectively. By the law of large numbers, $\widehat{\ell}(\mathbf{u}; \mathbf{v})$ converges to $\ell(\mathbf{u})$ with probability 1 (w.p.1) as $N \to \infty$. As shown in [19] under mild additional conditions, $\widehat{\ell}^*$ and $\widehat{\mathbf{u}}^*$ converge w.p.1 to their corresponding optimal objective value and to the optimal solution of the true problem (7.47), respectively. That is, $\widehat{\ell}^*$ and $\widehat{\mathbf{u}}^*$ are consistent estimators of their true counterparts ℓ^* and $\mathbf{u}^*$, respectively. Moreover, [19] establishes a central limit theorem and valid confidence regions for the tuple $(\ell^*, \mathbf{u}^*)$. The following theorem summarizes the basic statistical properties of $\widehat{\mathbf{u}}^*$ for the unconstrained program formulation. Additional discussion, including proofs for both the unconstrained and constrained programs, can be found in [19].

Theorem 7.3.2 *Let $\mathbf{u}^*$ be a unique minimizer of $\ell(\mathbf{u})$ over $\mathscr{V}$.*

A. Suppose that

1. *The set $\mathscr{V}$ is compact.*
2. *For almost every $\mathbf{x}$, the function $f(\mathbf{x}; \cdot)$ is continuous on $\mathscr{V}$.*
3. *The family of functions $\{|H(\mathbf{x})f(\mathbf{x}; \mathbf{u})|, \mathbf{u} \in \mathscr{V}\}$ is dominated by an integrable function $h(\mathbf{x})$, that is,*

$$|H(\mathbf{x})f(\mathbf{x}; \mathbf{u})| \leqslant h(\mathbf{x}) \quad \text{for all} \quad \mathbf{u} \in \mathscr{V}.$$

Then the optimal solution $\widehat{\mathbf{u}}^*$ of (7.48) converges to $\mathbf{u}^*$ as $N \to \infty$, with probability one.

B. Suppose further that

1. *$\mathbf{u}^*$ is an interior point of $\mathscr{V}$.*
2. *For almost every $\mathbf{x}, f(\mathbf{x}; \cdot)$ is twice continuously differentiable in a neighborhood $\mathscr{U}$ of $\mathbf{u}^*$, and the families of functions $\{\|H(\mathbf{x})\nabla^k f(\mathbf{x}; \mathbf{u})\| : \mathbf{u} \in \mathscr{U}, k = 1, 2\}$, where $\|\mathbf{x}\| = (x_1^2 + \cdots + x_n^2)^{\frac{1}{2}}$, are dominated by an integrable function.*
3. *The matrix*

$$B = \mathbb{E}_{\mathbf{v}} \left[H(\mathbf{X}) \nabla^2 W(\mathbf{X}; \mathbf{u}^*, \mathbf{v}) \right] \tag{7.52}$$

   is nonsingular.
4. *The covariance matrix of the vector $H(\mathbf{X})\nabla W(\mathbf{X}; \mathbf{u}^*, \mathbf{v})$, given by*

$$\Sigma = \mathbb{E}_{\mathbf{v}} \left[H^2(\mathbf{X}) \nabla W(\mathbf{X}; \mathbf{u}^*, \mathbf{v}) (\nabla W(\mathbf{X}; \mathbf{u}^*, \mathbf{v}))^\top \right] - \nabla \ell(\mathbf{u}^*) (\nabla \ell(\mathbf{u}^*))^\top ,$$

   exists.

Then the random vector $N^{1/2}(\widehat{\mathbf{u}}^* - \mathbf{u}^*)$ converges in distribution to a normal random vector with zero mean and covariance matrix

$$B^{-1} \Sigma B^{-1}. \tag{7.53}$$

The asymptotic efficiency of the estimator $N^{1/2}(\widehat{\mathbf{u}^*} - \mathbf{u}^*)$ is controlled by the covariance matrix given in (7.53). Under the assumptions of Theorem 7.3.2, this covariance matrix can be consistently estimated by $\widehat{B}^{-1}\widehat{\Sigma}\widehat{B}^{-1}$, where

$$\widehat{B} = \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i)\nabla^2 W(\mathbf{X}_i; \widehat{\mathbf{u}^*}, \mathbf{v}) \tag{7.54}$$

and

$$\widehat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N} H^2(\mathbf{X}_i)\,\nabla W(\mathbf{X}_i; \widehat{\mathbf{u}}, \mathbf{v})(\nabla W(\mathbf{X}_i; \widehat{\mathbf{u}^*}, \mathbf{v}))^\top - \widehat{\nabla\ell}(\widehat{\mathbf{u}^*}; \mathbf{v})(\widehat{\nabla\ell}(\widehat{\mathbf{u}^*}; \mathbf{v}))^\top \tag{7.55}$$

are consistent estimators of the matrices B and Σ, respectively. Observe that these matrices can be estimated from the same sample $\{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ simultaneously with the estimator $\widehat{\mathbf{u}^*}$. Observe also that the matrix B coincides with the Hessian matrix $\nabla^2\ell(\mathbf{u}^*)$ and is, therefore, symmetric and independent of the choice of the importance sampling parameter vector $\mathbf{v}$.

Although Theorem 7.3.2 was formulated for the distributional case only, similar arguments [19] apply to the stochastic counterpart (7.43), involving both distributional and structural parameter vectors $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively.

The statistical inference for the estimators $\widehat{\ell^*}$ and $\widehat{\mathbf{u}^*}$ allows the construction of stopping rules, validation analysis, and error bounds for the obtained solutions. In particular, it is shown in Shapiro [20] that if the function $\ell(\mathbf{u})$ is twice differentiable, then the above stochastic counterpart method produces estimators that converge to an optimal solution of the true problem at the same asymptotic rate as the stochastic approximation method, provided that the stochastic approximation method is applied with the *asymptotically optimal* step sizes. Moreover, it is shown in Kleywegt, Shapiro, and Homem de Mello [10] that if the underlying probability distribution is discrete and $\ell(\mathbf{u})$ is piecewise linear and convex, then w.p.1 the stochastic counterpart method (also called the *sample path method*) provides an exact optimal solution. For a recent survey on simulation-based optimization see Kleywegt and Shapiro [9].

The following example deals with unconstrained minimization of $\ell(\mathbf{u})$, where $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ and therefore contains both distributional and structural parameter vectors.

■ EXAMPLE 7.9 Examples 7.1 and 7.7 (Continued)

Consider minimization of the function

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X}; \mathbf{u}_2)] + \mathbf{b}^\top\mathbf{u}\,,$$

where

$$H(\mathbf{X}; u_3, u_4) = \max\{X_1 + u_3, X_2 + u_4\}\,, \tag{7.56}$$

$\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$, $\mathbf{u}_1 = (u_1, u_2)$, $\mathbf{u}_2 = (u_3, u_4)$, $\mathbf{X} = (X_1, X_2)$ is a two-dimensional vector with independent components, $X_i \sim f_i(x; u_i)$, $i = 1, 2$, with $X_i \sim \mathsf{Exp}(u_i)$, and $\mathbf{b} = (b_1, \ldots, b_4)$ is a cost vector.

To find the estimate of the optimal solution $\mathbf{u}^*$, we will use, as in Example 7.7, the direct, inverse-transform and push-out estimators of $\nabla\ell(\mathbf{u})$. In particular, we will define a system of nonlinear equations of type (7.44), which

is generated by the corresponding direct, inverse-transform, and push-out estimators of $\nabla \ell(\mathbf{u})$. Note that each such estimator will be associated with a proper likelihood ratio function $W(\cdot)$.

(a) *The direct estimator of* $\nabla \ell(\mathbf{u})$. In this case

$$W(\mathbf{X}; \mathbf{u}_1, \mathbf{v}_1) = \frac{f_1(X_1; u_1) f_2(X_2; u_2)}{f_1(X_1; v_1) f_2(X_2; v_2)},$$

where $\mathbf{X} \sim f_1(x_1; v_1) f_2(x_2; v_2)$ and $\mathbf{v}_1 = (v_1, v_2)$. Using the likelihood ratio term given above, we can rewrite formulas (7.31) and (7.32) as

$$\frac{\partial \ell(\mathbf{u})}{\partial u_1} = \mathbb{E}_{\mathbf{v}_1} \left[H(\mathbf{X}; \mathbf{u}_2) W(\mathbf{X}; \mathbf{u}_1, \mathbf{v}_1) \nabla \ln f_1(X_1; u_1) \right] + b_1 \tag{7.57}$$

and

$$\frac{\partial \ell(\mathbf{u})}{\partial u_3} = \mathbb{E}_{\mathbf{v}_1} \left[\frac{\partial H(\mathbf{X}; \mathbf{u}_2)}{\partial u_3} W(\mathbf{X}; \mathbf{u}_1, \mathbf{v}_1) \right] + b_3 \,, \tag{7.58}$$

respectively, and similarly $\partial \ell(\mathbf{u})/\partial u_2$ and $\partial \ell(\mathbf{u})/\partial u_4$. By analogy to (7.34), the importance sampling estimator of $\partial \ell(\mathbf{u})/\partial u_3$ can be written as

$$\widehat{\nabla \ell}_3^{(1)}(\mathbf{u}; \mathbf{v}_1) = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial H(\mathbf{X}_i; \mathbf{u}_2)}{\partial u_3} W(\mathbf{X}_i; \mathbf{u}_1, \mathbf{v}_1) + b_3 \,, \tag{7.59}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from $f(\mathbf{x}; \mathbf{v}_1) = f_1(x_1; v_1)\, f_2(x_2; v_2)$, and similarly for the remaining importance sampling estimators $\widehat{\nabla \ell}_i^{(1)}(\mathbf{u}; \mathbf{v}_1)$ of $\partial \ell(\mathbf{u})/\partial u_i,\ i = 1, 2, 4$. With this at hand, the estimate of the optimal solution $\mathbf{u}^*$ can be obtained from the solution of the following four-dimensional system of nonlinear equations:

$$\widehat{\nabla \ell}^{(1)}(\mathbf{u}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{V}, \tag{7.60}$$

where $\widehat{\nabla \ell}^{(1)} = (\widehat{\nabla \ell}_1^{(1)}, \ldots, \widehat{\nabla \ell}_4^{(1)})$.

(b) *The inverse-transform estimator of* $\nabla \ell(\mathbf{u})$. Taking (7.35) into account, the estimate of the optimal solution $\mathbf{u}^*$ can be obtained by solving, by analogy to (7.60), the following four-dimensional system of nonlinear equations:

$$\widehat{\nabla \ell}^{(2)}(\mathbf{u}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{V} \,. \tag{7.61}$$

Here, as before,

$$\widehat{\nabla \ell}^{(2)}(\mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} \nabla \breve{H}(\mathbf{Z}_i; \mathbf{u}) + \mathbf{b} \,,$$

and $\mathbf{Z}_1, \ldots, \mathbf{Z}_N$ is a random sample from the two-dimensional uniform pdf with independent components, that is, $\mathbf{Z} = (Z_1, Z_2)$ and $Z_j \sim \mathsf{U}(0, 1)$, $j = 1, 2$. Alternatively, we could estimate $\mathbf{u}^*$ using the ITLR method. In this case, by analogy to (7.61), the four-dimensional system of nonlinear equations can be written as

$$\widehat{\nabla \ell}^{(2)}(\mathbf{u}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{V} \,, \tag{7.62}$$

with

$$\widetilde{\nabla\ell}^{(2)}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} \nabla\breve{H}(\mathbf{X}_i;\mathbf{u})W(\mathbf{X}_i;\boldsymbol{\theta}) + \mathbf{b}\ ,$$

and

$$W(\mathbf{X}_i;\boldsymbol{\theta}) = \frac{1}{h_1(X_{1i};\theta_1)\,h_2(X_{2i};\theta_2)},$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2)$, $\mathbf{X} = (X_1, X_2) \sim h_1(x_1;\theta_1)h_2(x_2;\theta_2)$, and, for example, $h_i(x;\theta_i) = \theta_i x^{\theta_i - 1}$, $i = 1, 2$; that is, $h_i(\cdot)$ is a Beta pdf.

(c) *The push-out estimator of* $\nabla\ell(\mathbf{u})$. Using (7.39), the estimate of the optimal solution $\mathbf{u}^*$ can be obtained from the solution of the following four-dimensional system of nonlinear equations:

$$\widehat{\nabla\ell}^{(3)}(\mathbf{u};\mathbf{v}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{V}, \tag{7.63}$$

where

$$\widehat{\nabla\ell}^{(3)}(\mathbf{u};\mathbf{v}) = \frac{1}{N}\sum_{i=1}^{N} \max\{\widetilde{X}_{1i}, \widetilde{X}_{2i}\}\, W(\widetilde{\mathbf{X}}_i;\mathbf{u},\mathbf{v})\nabla \ln \widetilde{f}(\widetilde{\mathbf{X}}_i;\mathbf{u}) + \mathbf{b}\ ,$$

$$W(\widetilde{\mathbf{X}}_i;\mathbf{u},\mathbf{v}) = \frac{f_1(X_1 - u_3;u_1)\, f_2(X_2 - u_4;u_2)}{f_1(X_1 - v_3;v_1)\, f_2(X_2 - v_4;v_2)}$$

and $\widetilde{\mathbf{X}}_i \sim \widetilde{f}(\mathbf{x}) = f_1(x_1 - v_3;v_1)\, f_2(x_2 - v_4;v_2)$.

Let us return finally to the stochastic counterpart of the general program (P_0). From the foregoing discussion, it can be written as

$$(\widehat{\mathrm{P}}_N) \qquad \begin{array}{lll} \text{minimize} & \widehat{\ell}_0(\mathbf{u};\mathbf{v}_1), & \mathbf{u} \in \mathscr{V}, \\ \text{subject to:} & \widehat{\ell}_j(\mathbf{u};\mathbf{v}_1) \leqslant 0, & j = 1,\dots,k, \\ & \widehat{\ell}_j(\mathbf{u};\mathbf{v}_1) = 0, & j = k+1,\dots,M, \end{array} \tag{7.64}$$

with

$$\widehat{\ell}_j(\mathbf{u};\mathbf{v}_1) = \frac{1}{N}\sum_{i=1}^{N} H_j(\mathbf{X}_i;\mathbf{u}_2)\, W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1), \quad j = 0, 1, \dots, M, \tag{7.65}$$

where $\mathbf{X}_1, \dots, \mathbf{X}_N$ is a random sample from the importance sampling pdf $f(\mathbf{x};\mathbf{v}_1)$, and the $\{\widehat{\ell}_j(\mathbf{u};\mathbf{v}_1)\}$ are viewed as functions of $\mathbf{u}$ rather than as estimators for a fixed $\mathbf{u}$.

Note again that once the sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ is generated, the functions $\widehat{\ell}_j(\mathbf{u};\mathbf{v}_1)$, $j = 0, \dots, M$, become *explicitly* determined via the functions $H_j(\mathbf{X}_i;\mathbf{u}_2)$ and $W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1)$. Let us assume that the corresponding gradients $\nabla\widehat{\ell}_j(\mathbf{u};\mathbf{v}_1)$ can be calculated, for any $\mathbf{u}$, from a single simulation run, so that we can solve the optimization problem $(\widehat{\mathrm{P}}_N)$ by standard methods of mathematical programming. The resultant optimal function value and the optimal decision vector of the program $(\widehat{\mathrm{P}}_N)$ provide estimators of the optimal values ℓ^* and $\mathbf{u}^*$, respectively, of the original one (P_0). It is important to understand that what makes this approach

feasible is the fact that once the sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is generated, the functions $\widehat{\ell}_j(\mathbf{u})$, $j = 0, \ldots, M$ become known explicitly, provided that the sample functions $\{H_j(\mathbf{X}; \mathbf{u}_2)\}$ are explicitly available for any $\mathbf{u}_2$. Recall that if $H_j(\mathbf{X}; \mathbf{u}_2)$ is available only for some fixed in advance $\mathbf{u}_2$, rather than simultaneously for all values $\mathbf{u}_2$, then stochastic approximation algorithms can be applied instead of the stochastic counterpart method. Note that in the case where the $\{H_j(\cdot)\}$ do not depend on $\mathbf{u}_2$, one can solve the program $(\widehat{\mathrm{P}}_N)$ (from a single simulation run) using the SF method, provided that the trust region of the program $(\widehat{\mathrm{P}}_N)$ does not exceed the one defined in (7.27). If this is not the case, iterative gradient-type methods need to be used, since they do not involve likelihood ratios.

The algorithm for estimating the optimal solution, $\mathbf{u}^*$, of the program (P_0) via the stochastic counterpart $(\widehat{\mathrm{P}}_N)$ can be written as follows:

Algorithm 7.3.2: Estimation of $\mathbf{u}^*$

1 Generate a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{v}_1)$.
2 Calculate the functions $H_j(\mathbf{X}_i; \mathbf{u}_2)$, $j = 0, \ldots, M$, $i = 1, \ldots, N$ via simulation.
3 Solve the program $(\widehat{\mathrm{P}}_N)$ by standard mathematical programming methods.
4 Return the resultant optimal solution, $\widehat{\mathbf{u}^*}$ of $(\widehat{\mathrm{P}}_N)$, as an estimate of $\mathbf{u}^*$.

The third step of Algorithm 7.3.2 typically calls for iterative numerical procedures, which may require, in turn, calculation of the functions $\widehat{\ell}_j(\mathbf{u})$, $j = 0, \ldots, M$, and their gradients (and possibly Hessians), for multiple values of the parameter vector $\mathbf{u}$. Our extensive simulation studies for typical DESS with sizes up to 100 decision variables show that the optimal solution $\widehat{\mathbf{u}^*}$ of the program $(\widehat{\mathrm{P}}_N)$ constitutes a reliable estimator of the true optimal solution, $\mathbf{u}^*$, provided that the program $(\widehat{\mathrm{P}}_N)$ is convex (see [19] and the Appendix), the trust region is not too large, and the sample size N is quite large (on the order of 1000 or more).

7.4 SENSITIVITY ANALYSIS OF DEDS

Let $X_1, X_2, \ldots$ be an input sequence of m-dimensional random vectors driving an output process $\{H_t, t = 0, 1, 2, \ldots\}$. That is, $H_t = H_t(\mathbf{X}_t)$ for some function H_t, where the vector $\mathbf{X}_t = (X_1, X_2, \ldots, X_t)$ represents the history of the input process up to time t. Let the pdf of $\mathbf{X}_t$ be given by $f_t(\mathbf{x}_t; \mathbf{u})$, which depends on some parameter vector $\mathbf{u}$. Assume that $\{H_t\}$ is a regenerative process with a regenerative cycle of length τ. Typical examples are an ergodic Markov chain and the waiting time process in the $GI/G/1$ system. In both cases (see Section 4.4.2.2) the expected steady-state performance, $\ell(\mathbf{u})$, can be written as

$$\ell(\mathbf{u}) = \frac{\mathbb{E}_{\mathbf{u}}[R]}{\mathbb{E}_{\mathbf{u}}[\tau]} = \frac{\mathbb{E}_{\mathbf{u}}\left[\sum_{t=1}^{\tau} H_t(\mathbf{X}_t)\right]}{\mathbb{E}_{\mathbf{u}}[\tau]}\,, \tag{7.66}$$

where R is the reward during a cycle. As for static models, we show here how to estimate from a *single simulation run* the performance $\ell(\mathbf{u})$, and the derivatives $\nabla^k \ell(\mathbf{u})$, $k = 1, 2 \ldots$, for different values of $\mathbf{u}$.

Consider first the estimation of $\ell_R(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[R]$ when the $\{X_i\}$ are iid with pdf $f(x; \mathbf{u})$; thus, $f_t(\mathbf{x}_t) = \prod_{i=1}^{t} f(x_i)$. Let $g(x)$ be any importance sampling pdf, and

let $g_t(\mathbf{x}_t) = \prod_{i=1}^{t} g(x_i)$. It will be shown that $\ell_R(\mathbf{u})$ can be represented as

$$\ell_R(\mathbf{u}) = \mathbb{E}_g \left[\sum_{t=1}^{\tau} H_t(\mathbf{X}_t)\, W_t(\mathbf{X}_t; \mathbf{u}) \right], \tag{7.67}$$

where $\mathbf{X}_t \sim g_t(\mathbf{x}_t)$ and $W_t(\mathbf{X}_t; \mathbf{u}) = f_t(\mathbf{x}_t; \mathbf{u})/g_t(\mathbf{x}_t) = \prod_{j=1}^{t} f(X_j; \mathbf{u})/g(X_j)$. To proceed, we write

$$\sum_{t=1}^{\tau} H_t = \sum_{t=1}^{\infty} H_t\, I_{\{\tau \geqslant t\}} \,. \tag{7.68}$$

Since $\tau = \tau(\mathbf{X}_t)$ is completely determined by $\mathbf{X}_t$, the indicator $I_{\{\tau \geqslant t\}}$ can be viewed as a function of $\mathbf{x}_t$; we write $I_{\{\tau \geqslant t\}}(\mathbf{x}_t)$. Accordingly, the expectation of $H_t\, I_{\{\tau \geqslant t\}}$ is

$$\begin{aligned} \mathbb{E}_{\mathbf{u}}[H_t\, I_{\{\tau \geqslant t\}}] &= \int H_t(\mathbf{x}_t)\, I_{\{\tau \geqslant t\}}(\mathbf{x}_t)\, f_t(\mathbf{x}_t; \mathbf{u})\, \mathrm{d}\mathbf{x}_t \\ &= \int H_t(\mathbf{x}_t)\, I_{\{\tau \geqslant t\}}(\mathbf{x}_t)\, W_t(\mathbf{x}_t; \mathbf{u})\, g_t(\mathbf{x}_t)\, \mathrm{d}\mathbf{x}_t \\ &= \mathbb{E}_g[H_t(\mathbf{X}_t)\, I_{\{\tau \geqslant t\}}(\mathbf{X}_t)\, W_t(\mathbf{X}_t; \mathbf{u})] \,. \end{aligned} \tag{7.69}$$

The result (7.67) follows by combining (7.68) and (7.69). For the special case where $H_t \equiv 1$, (7.67) reduces to

$$\mathbb{E}_{\mathbf{u}}[\tau] = \mathbb{E}_g \left[\sum_{t=1}^{\tau} W_t \right],$$

abbreviating $W_t(\mathbf{X}_t; \mathbf{u})$ to W_t. Derivatives of (7.67) can be presented in a similar form. In particular, under standard regularity conditions ensuring the interchangeability of the differentiation and the expectation operators, one can write

$$\nabla^k \ell_R(\mathbf{u}) = \mathbb{E}_g \left[\sum_{t=1}^{\tau} H_t \nabla^k W_t \right] = \mathbb{E}_g \left[\sum_{t=1}^{\tau} H_t\, \mathcal{S}_t^{(k)}\, W_t \right] \,, \tag{7.70}$$

where $\mathcal{S}_t^{(k)}$ is the k-th order score function corresponding to $f_t(\mathbf{x}_t; \mathbf{u})$, as in (7.7).

Now let $\{X_{11}, \ldots, X_{\tau_1 1}, \ldots, X_{1N}, \ldots, X_{\tau_N N}\}$ be a sample of N regenerative cycles from the pdf $g(x)$. Then, using (7.70), we can estimate $\nabla^k \ell_R(\mathbf{u})$, $k = 0, 1, \ldots$ from a *single simulation run* as

$$\widehat{\nabla^k \ell_R}(\mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{\tau_i} H_{ti}\, \mathcal{S}_{ti}^{(k)}\, W_{ti} \,, \tag{7.71}$$

where $W_{ti} = \prod_{j=1}^{t} \frac{f(X_{ji}; \mathbf{u})}{g(X_{ji})}$ and $X_{ji} \sim g(x)$. Notice that here $\widehat{\nabla^k \ell_R}(\mathbf{u}) = \nabla^k \widehat{\ell_R}(\mathbf{u})$. For the special case where $g(\mathbf{x}) = f(\mathbf{x}; \mathbf{u})$, that is, when using the original pdf $f(\mathbf{x}; \mathbf{u})$, one has

$$\nabla^k \ell_R(\mathbf{u}) = \mathbb{E}_{\mathbf{u}} \left[\sum_{t=1}^{\tau} H_t\, \mathcal{S}_t^{(k)} \right]. \tag{7.72}$$

For $k = 1$, writing $\mathcal{S}_t$ for $\mathcal{S}_t^{(1)}$, the score function process $\{\mathcal{S}_t\}$ is given by

$$\mathcal{S}_t = \sum_{j=1}^{t} \nabla \ln f(X_j; \mathbf{u}) \,. \tag{7.73}$$

■ **EXAMPLE 7.10**

Let $X \sim \mathsf{G}(p)$. That is, $f(x;p) = p\,(1-p)^{x-1}$, $x = 1, 2, \ldots$. Then (see also Table 7.1)

$$\mathcal{S}_t = \frac{\partial}{\partial p} \ln f_t(\mathbf{X}_t; p) = \frac{t - p \sum_{j=1}^{t} X_j}{p(1-p)} \,.$$

■ **EXAMPLE 7.11**

Let $X \sim \mathsf{Gamma}(\alpha, \lambda)$. That is, $f(x;\lambda,\alpha) = \frac{\lambda^\alpha x^{\alpha-1} \mathrm{e}^{-\lambda x}}{\Gamma(\alpha)}$ for $x > 0$. Suppose we are interested in the sensitivities with respect to λ. Then

$$\mathcal{S}_t = \frac{\partial}{\partial \lambda} \ln f_t(\mathbf{X}_t; \lambda, \alpha) = t\,\alpha\,\lambda^{-1} - \sum_{i=1}^{t} X_i \,.$$

Let us return now to the estimation of $\ell(\mathbf{u}) = \mathbb{E}_\mathbf{u}[R]/\mathbb{E}_\mathbf{u}[\tau]$ and its sensitivities. In view of (7.70) and the fact that $\tau = \sum_{t=1}^{\tau} 1$ can be viewed as a special case of (7.67), with $H_t = 1$, one can write $\ell(\mathbf{u})$ as

$$\ell(\mathbf{u}) = \frac{\mathbb{E}_g[\sum_{t=1}^{\tau} H_t\, W_t]}{\mathbb{E}_g[\sum_{t=1}^{\tau} W_t]} \tag{7.74}$$

and by direct differentiation of (7.74) write $\nabla\ell(\mathbf{u})$ as

$$\nabla\ell(\mathbf{u}) = \frac{\mathbb{E}_g[\sum_{t=1}^{\tau} H_t\, \nabla W_t]}{\mathbb{E}_g[\sum_{t=1}^{\tau} W_t]} - \frac{\mathbb{E}_g[\sum_{t=1}^{\tau} H_t\, W_t]}{\mathbb{E}_g[\sum_{t=1}^{\tau} W_t]} \cdot \frac{\mathbb{E}_g[\sum_{t=1}^{\tau} \nabla W_t]}{\mathbb{E}_g[\sum_{t=1}^{\tau} W_t]} \tag{7.75}$$

(observe that $W_t = W_t(\mathbf{X}_t, \mathbf{u})$ is a function of $\mathbf{u}$ but $H_t = H_t(\mathbf{X}_t)$ is not). Observe also that above, $\nabla W_t = W_t\, \mathcal{S}_t$. Higher-order partial derivatives with respect to parameters of interest can then be obtained from (7.75). Utilizing (7.74) and (7.75), one can estimate $\ell(\mathbf{u})$ and $\nabla\ell(\mathbf{u})$, for all $\mathbf{u}$, as

$$\widehat{\ell}(\mathbf{u}) = \frac{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} H_{ti}\, W_{ti}}{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti}} \tag{7.76}$$

and

$$\widehat{\nabla\ell}(\mathbf{u}) = \frac{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} H_{ti} W_{ti} \mathcal{S}_{ti}}{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti}} - \frac{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} H_{ti} W_{ti}}{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti}} \cdot \frac{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti} \mathcal{S}_{ti}}{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti}} \,, \tag{7.77}$$

respectively, and similarly for higher-order derivatives. Notice again that in this case, $\widehat{\nabla\ell}(\mathbf{u}) = \nabla\widehat{\ell}(\mathbf{u})$. The algorithm for estimating the gradient $\nabla\ell(\mathbf{u})$ at different values of $\mathbf{u}$ using a single simulation run can be written as follows:

Algorithm 7.4.1: $\nabla\ell(\mathbf{u})$ Estimation

1 Generate a random sample $\{X_1, \ldots, X_T\}$, $T = \sum_{i=1}^{N} \tau_i$, from $g(x)$.
2 Generate the output processes $\{H_t\}$ and $\{\nabla W_t\} = \{W_t \mathcal{S}_t\}$.
3 Calculate $\widehat{\nabla\ell}(\mathbf{u})$ from (7.77).

Confidence intervals (regions) for the sensitivities $\nabla^k \ell(\mathbf{u}),\ k = 0, 1$, utilizing the SF estimators $\nabla^k \widehat{\ell}(\mathbf{u}),\ k = 0, 1$, can be derived analogously to those for the standard regenerative estimator of Chapter 4 and are left as an exercise.

■ **EXAMPLE 7.12 Waiting Time**

The waiting time process in a $GI/G/1$ queue is driven by sequences of inter-arrival times $\{A_t\}$ and service times $\{S_t\}$ via the Lindley equation

$$H_t = \max\{H_{t-1} + S_t - A_t,\ 0\}, \quad t = 1, 2, \ldots, \tag{7.78}$$

with $H_0 = 0$; see (4.33) and Problem 5.3. Writing $X_t = (S_t, A_t)$, the $\{X_t, t = 1, 2, \ldots\}$ are iid. The process $\{H_t, t = 0, 1, \ldots\}$ is a regenerative process, which regenerates every time $H_t = 0$. Let $\tau > 0$ denote the first such time, and let H denote the steady-state waiting time. We wish to estimate the steady-state performance

$$\ell = \mathbb{E}[H] = \frac{\mathbb{E}\left[\sum_{t=1}^{\tau} H_t\right]}{\mathbb{E}[\tau]} .$$

Consider, for instance, the case where $S \sim \mathsf{Exp}(\mu)$, $A \sim \mathsf{Exp}(\lambda)$, and S and A are independent. Thus, H is the steady-state waiting time in the $M/M/1$ queue, and $\mathbb{E}[H] = \lambda/(\mu(\mu - \lambda))$ for $\mu > \lambda$; see, for example, [6]. Suppose we carry out the simulation using the service rate $\tilde{\mu}$ and wish to estimate $\ell(\mu) = \mathbb{E}[H]$ for different values of μ using the same simulation run. Let $(S_1, A_1), \ldots, (S_\tau, A_\tau)$ denote the service and inter-arrival times in the first cycle, respectively. Then, for the first cycle

$$W_t = W_{t-1} \frac{\mu\, \mathrm{e}^{-\mu S_t}}{\tilde{\mu}\, \mathrm{e}^{-\tilde{\mu} S_t}}, \quad t = 1, 2, \ldots, \tau \quad (W_0 = 1) ,$$

$$\mathcal{S}_t = \mathcal{S}_{t-1} + \frac{1}{\mu} - S_t, \quad t = 1, 2, \ldots, \tau \quad (\mathcal{S}_0 = 0) ,$$

and H_t is as given in (7.78). From these, the sums $\sum_{t=1}^{\tau} H_t W_t$, $\sum_{t=1}^{\tau} W_t$, $\sum_{t=1}^{\tau} W_t \mathcal{S}_t$, and $\sum_{t=1}^{\tau} H_t W_t \mathcal{S}_t$ can be computed. Repeating this for the subsequent cycles, we can estimate $\ell(\mu)$ and $\nabla \ell(\mu)$ from (7.76) and (7.77), respectively. Figure 7.4 displays the estimates and true values for $1.5 \leqslant \mu \leqslant 5.5$ using a single simulation run of $N = 10^5$ cycles. The simulation was carried out under the service rate $\tilde{\mu} = 2$ and arrival rate $\lambda = 1$. We see that both $\ell(\mu)$ and $\nabla \ell(\mu)$ are estimated accurately over the whole range. Note that for $\mu < 2$ the confidence interval for $\ell(\mu)$ grows rapidly wider. The estimation should not be extended much below $\mu = 1.5$, as the importance sampling will break down, resulting in unreliable estimates.

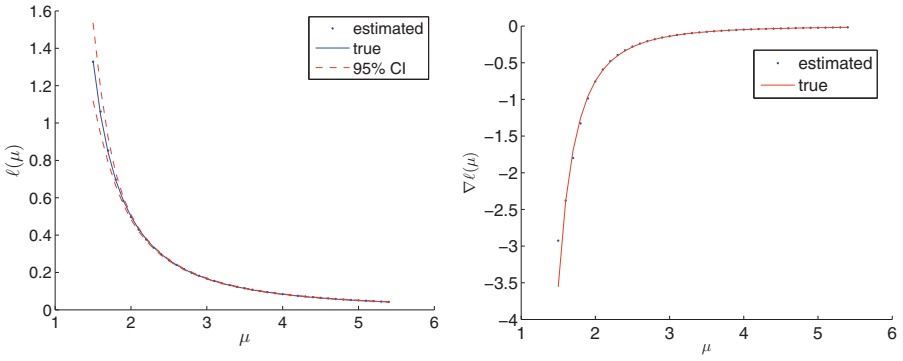

Figure 7.4: Estimated and true values for the expected steady-state waiting time and its derivative as a function of μ.

Although (7.76) and (7.77) were derived for the case where the $\{X_i\}$ are iid, much of the theory can be readily modified to deal with the dependent case. As an example, consider the case where $X_1, X_2, \ldots$ form an ergodic Markov chain and R is of the form

$$R = \sum_{t=1}^{\tau} c_{X_{t-1},X_t} \;, \tag{7.79}$$

where c_{ij} is the cost of going from state i to j and R represents the cost accrued in a cycle of length τ. Let $\mathbf{P} = (p_{ij})$ be the one-step transition matrix of the Markov chain. Following reasoning similar to that for (7.67) and defining $H_t = c_{X_{t-1},X_t}$, we see that

$$\mathbb{E}_{\mathbf{P}}[R] = \mathbb{E}_{\widetilde{\mathbf{P}}}\left[\sum_{t=1}^{\tau} H_t\, W_t\right],$$

where $\widetilde{\mathbf{P}} = (\widetilde{p}_{ij})$ is another transition matrix, and

$$W_t = W_t(\mathbf{X}_t; \mathbf{P}, \widetilde{\mathbf{P}}) = \prod_{k=1}^{t} \frac{p_{X_{k-1},X_k}}{\widetilde{p}_{X_{k-1},X_k}}$$

is the likelihood ratio. The pdf of $\mathbf{X}_t$ is given by

$$f_t(\mathbf{x}_t; \mathbf{P}) = \prod_{k=1}^{t} p_{X_{k-1},X_k} \;.$$

The score function can again be obtained by taking the derivative of the logarithm of the pdf. Since, $\mathbb{E}_{\mathbf{P}}[\tau] = \mathbb{E}_{\widetilde{\mathbf{P}}}[\sum_{t=1}^{\tau} W_t]$, the long-run average cost $\ell(\mathbf{P}) = \mathbb{E}_{\mathbf{P}}[R]/\mathbb{E}_{\mathbf{P}}[\tau]$ can be estimated via (7.76) — and its derivatives by (7.77) — simultaneously for various $\mathbf{P}$ using a single simulation run under $\widetilde{\mathbf{P}}$.

EXAMPLE 7.13 Markov Chain: Example 4.9 (Continued)

Consider again the two-state Markov chain with transition matrix $\mathbf{P} = (p_{ij})$ and cost matrix C given by

$$\mathbf{P} = \begin{pmatrix} p_1 & 1-p_1 \\ p_2 & 1-p_2 \end{pmatrix} = \begin{pmatrix} \mathbf{p} & \mathbf{1}-\mathbf{p} \end{pmatrix}$$

and

$$C = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix},$$

respectively, where $\mathbf{p}$ denotes the vector $(p_1, p_2)^\top$. Our goal is to estimate $\ell(\mathbf{p})$ and $\nabla\ell(\mathbf{p})$ using (7.76) and (7.77) for various $\mathbf{p}$ from a single simulation run under $\widetilde{\mathbf{p}} = (\frac{1}{2}, \frac{1}{5})^\top$. Assume, as in Example 4.9 on Page 124, that starting from state 1, we obtain the sample trajectory $(x_0, x_1, x_2, \ldots, x_{10}) = (1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 1)$, which has four cycles with lengths $\tau_1 = 4,\ \tau_2 = 2,\ \tau_3 = 1,\ \tau_4 = 3$, and corresponding transition probabilities $(p_{12}, p_{22}, p_{22}, p_{21})$; (p_{12}, p_{21}); (p_{11}); (p_{12}, p_{22}, p_{21}). The cost in the first cycle is given by (7.79). We consider the cases (1) $\mathbf{p} = \widetilde{\mathbf{p}} = (\frac{1}{2}, \frac{1}{5})^\top$ and (2) $\mathbf{p} = (\frac{1}{5}, \frac{1}{2})^\top$. The transition matrices for the two cases are

$$\widetilde{\mathbf{P}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{5} & \frac{4}{5} \end{pmatrix} \qquad \text{and} \qquad \mathbf{P} = \begin{pmatrix} \frac{1}{5} & \frac{4}{5} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

Note that the first case pertains to the nominal Markov chain.

In the first cycle, costs $H_{11} = 1, H_{21} = 3, H_{31} = 3$, and $H_{41} = 2$ are incurred. The likelihood ratios under case (2) are $W_{11} = \frac{p_{12}}{\widetilde{p}_{12}} = 8/5$, $W_{21} = W_{11}\frac{p_{22}}{\widetilde{p}_{22}} = 1$, $W_{31} = W_{21}\frac{p_{22}}{\widetilde{p}_{22}} = \frac{5}{8}$, and $W_{41} = W_{31}\frac{p_{21}}{\widetilde{p}_{21}} = \frac{25}{16}$, while in case (1) they are all 1. Next, we derive the score functions (in the first cycle) with respect to p_1 and p_2. Note that

$$f_4(\mathbf{x}_4; \mathbf{p}) = p_{12}\, p_{22}^2\, p_{21} = (1-p_1)(1-p_2)^2\, p_2\,.$$

It follows that in case (2),

$$\frac{\partial}{\partial p_1} \ln f_4(\mathbf{x}_4; \mathbf{p}) = \frac{-1}{1-p_1} = -\frac{5}{4}$$

and

$$\frac{\partial}{\partial p_2} \ln f_4(\mathbf{x}_4; \mathbf{p}) = \frac{-2}{1-p_2} + \frac{1}{p_2} = -2\,,$$

so that the score function at time $t = 4$ in the first cycle is given by $\mathcal{S}_{41} = (-\frac{5}{4}, -2)$. Similarly, $\mathcal{S}_{31} = (-\frac{5}{4}, -4)$, $\mathcal{S}_{21} = (-\frac{5}{4}, -2)$, and $\mathcal{S}_{11} = (-\frac{5}{4}, 0)$. The quantities for the other cycles are derived in the same way, and the results are summarized in Table 7.3.

Table 7.3: Summary of costs, likelihood ratios, and score functions.

i	H_{ti}	W_{ti} (case 2)	$\mathcal{S}_{ti}$ (case 1)	$\mathcal{S}_{ti}$ (case 2)
1	1, 3, 3, 2	$\frac{8}{5}$, 1, $\frac{5}{8}$, $\frac{25}{16}$	$(-2,0)$, $(-2,-\frac{5}{4})$, $(-2,-\frac{5}{2})$, $(-2,\frac{5}{2})$	$(-\frac{5}{4},0)$, $(-\frac{5}{4},-2)$, $(-\frac{5}{4},-4)$, $(-\frac{5}{4},-2)$
2	1, 2	$\frac{8}{5}$, 4	$(-2,0)$,$(-2,5)$	$(-\frac{5}{4},0)$, $(-\frac{5}{4},2)$
3	0	$\frac{2}{5}$	$(2,0)$	$(5,0)$
4	1, 3, 2	$\frac{8}{5}$, 1, $\frac{5}{2}$	$(-2,0)$, $(-2,-\frac{5}{4})$, $(-2,\frac{15}{4})$	$(-\frac{5}{4},0)$, $(-\frac{5}{4},-2)$, $(-\frac{5}{4},0)$

By substituting these values in (7.76) and (7.77), the reader can verify that $\widehat{\ell}(\tilde{\mathbf{p}}) = 1.8$, $\widehat{\ell}(\mathbf{p}) \approx 1.81$, $\widehat{\nabla\ell}(\tilde{\mathbf{p}}) = (-0.52, -0.875)$, and $\widehat{\nabla\ell}(\mathbf{p}) \approx (0.22, -1.23)$.

PROBLEMS

7.1 Consider the unconstrained minimization program

$$\min_u \ell(u) = \min_u \left\{ \mathbb{E}_u[X] + \frac{b}{u} \right\}, \quad u \in (0,1) \,, \tag{7.80}$$

where $X \sim \mathsf{Ber}(u)$.

a) Show that the stochastic counterpart of $\nabla\ell(u) = 0$ can be written (see (7.18)) as

$$\widehat{\nabla\ell}(u) = \nabla\widehat{\ell}(u;v) = \frac{1}{v}\frac{1}{N}\sum_{i=1}^{N} X_i - \frac{b}{u^2} = 0 \,, \tag{7.81}$$

where $X_1, \ldots, X_N$ is a random sample from $\mathsf{Ber}(v)$.

b) Assume that the sample $\{0,1,0,0,1,0,0,1,1,1,0,1,0,1,1,0,1,0,1,1\}$ was generated from $\mathsf{Ber}(v = 1/2)$. Show that the optimal solution u^* is estimated as

$$\widehat{u^*} = \left(\frac{b}{1.1}\right)^{1/2} .$$

7.2 Consider the unconstrained minimization program

$$\min_u \ell(u) = \min_u \{\mathbb{E}_u[X] + bu\}, \;\; u \in (0.5, 2.0) \,, \tag{7.82}$$

where $X \sim \mathsf{Exp}(u)$. Show that the stochastic counterpart of $\nabla\ell(u) = -\frac{1}{u^2} + b = 0$ can be written (see (7.20)) as

$$\nabla\widehat{\ell}(u;v) = \frac{1}{N}\sum_{i=1}^{N} X_i \frac{\mathrm{e}^{-uX_i}(1 - u\,X_i)}{v\,\mathrm{e}^{-vX_i}} + b = 0 \,, \tag{7.83}$$

where $X_1, \ldots, X_n$ is a random sample from $\mathsf{Exp}(v)$.

7.3 Prove (7.25).

7.4 Show that $\nabla^k W(\mathbf{x};\mathbf{u},\mathbf{v}) = \mathcal{S}^{(k)}(\mathbf{u};\mathbf{x})\, W(\mathbf{x};\mathbf{u},\mathbf{v})$ and hence prove (7.16).

7.5 Let $X_i \sim \mathsf{N}(u_i, \sigma_i^2)$, $i = 1, \ldots, n$ be independent random variables. Here we are interested in sensitivities with respect to $\mathbf{u} = (u_1, \ldots, u_n)$ only. Show that, for $i = 1, \ldots, n$,

$$\mathbb{E}_{\mathbf{v}}[W^2] = \exp\left(\sum_{i=1}^{n} \frac{(u_i - v_i)^2}{\sigma_i^2}\right)$$

and

$$[\mathcal{S}^{(1)}(\mathbf{u}; \mathbf{X})]_i = \sigma_i^{-2}(x_i - u_i) \; .$$

7.6 Let the components X_i, $i = 1, \ldots, n$, of a random vector $\mathbf{X}$ be independent, and distributed according to the exponential family

$$f_i(x_i; \mathbf{u}_i) = c_i(\mathbf{u}_i)\, \mathrm{e}^{b_i(\mathbf{u}_i) t_i(x_i)}\, h_i(x_i) \; ,$$

where $b_i(\mathbf{u}_i)$, $t_i(x_i)$, and $h_i(x_i)$ are real-valued functions and $c_i(\mathbf{u}_i)$ is normalization constant. The corresponding pdf of $\mathbf{X}$ is given by

$$f(\mathbf{x}; \mathbf{u}) = c(\mathbf{u}) \exp\left(\sum_{i=1}^{n} b_i(\mathbf{u}_i) t_i(x_i)\right) h(\mathbf{x}) \; ,$$

where $\mathbf{u} = (\mathbf{u}_1^\top, \ldots, \mathbf{u}_n^\top)$, $c(\mathbf{u}) = \prod_{i=1}^n c_i(\mathbf{u}_i)$, and $h(\mathbf{x}) = \prod_{i=1}^n h_i(x_i)$.

a) Show that $\mathrm{Var}_{\mathbf{v}}(HW) = \frac{c(\mathbf{u})^2}{c(\mathbf{v})\, c(\mathbf{w})}\, \mathbb{E}_{\mathbf{w}}[H^2] - \ell(\mathbf{u})^2$, where $\mathbf{w}$ is determined by $b_i(\mathbf{w}_i) = 2b_i(\mathbf{u}_i) - b_i(\mathbf{v}_i)$, $i = 1, \ldots, n$.

b) Show that

$$\mathbb{E}_{\mathbf{v}}[H^2 W^2] = \mathbb{E}_{\mathbf{v}}[W^2]\, \mathbb{E}_{\mathbf{w}}[H^2] \; .$$

7.7 Consider the exponential pdf $f(x; u) = u \exp(-ux)$. Show that if $H(x)$ is a monotonically increasing function, then the expected performance $\ell(u) = \mathbb{E}_u[H(X)]$ is a monotonically decreasing convex function of $u \in (0, \infty)$.

7.8 Let $X \sim \mathsf{N}(u, \sigma^2)$. Suppose that σ is known and fixed. For a given u, consider the function

$$\mathcal{L}(v) = \mathbb{E}_v[H^2 W^2] \; .$$

a) Show that if $\mathbb{E}_u[H^2] < \infty$ for all $u \in \mathbb{R}$, then $\mathcal{L}(v)$ is convex and continuous on $\mathbb{R}$. Show further that if, additionally, $\mathbb{E}_{u_n}[H^2] > 0$ for any u, then $\mathcal{L}(v)$ has a unique minimizer, v^*, over $\mathbb{R}$.

b) Show that if $H^2(x)$ is monotonically increasing on $\mathbb{R}$, then $v^* > u$.

7.9 Let $X \sim \mathsf{N}(u, \sigma^2)$. Suppose that u is known, and consider the parameter σ. Note that the resulting exponential family is not of canonical form (A.9). However, parameterizing it by $\theta = \sigma^{-2}$ transforms it into canonical form, with $t(x) = -(x - u)^2/2$ and $c(\theta) = (2\pi)^{-1/2}\theta^{1/2}$.

a) Show that

$$\mathbb{E}_\eta[W^2] = \frac{\theta}{\eta^{1/2}(2\theta - \eta)^{1/2}} \; ,$$

provided that $0 < \eta < 2\theta$.

b) Show that, for a given θ, the function

$$\mathcal{L}(\eta) = \mathbb{E}_\eta[H^2 W^2]$$

has a unique minimizer, η^*, on the interval $(0, 2\theta)$, provided that the expectation, $\mathbb{E}_\eta[H^2]$, is finite for all $\eta \in (0, 2\theta)$ and does not tend to 0 as η approaches 0 or 2θ. (Notice that this implies that the corresponding optimal value, $\sigma^* = \eta^{*-1/2}$, of the reference parameter, σ, is also unique.)

c) Show that if $H^2(x)$ is strictly convex on $\mathbb{R}$, then $\eta^* < \theta$. (Notice that this implies that $\sigma^* > \sigma$.)

7.10 Consider the performance

$$H(X_1, X_2; u_3, u_4) = \min\{\max(X_1, u_3), \max(X_2, u_4)\} ,$$

where X_1 and X_2 have continuous densities $f(x_1; u_1)$ and $f(x_2; u_2)$, respectively. If we let $Y_1 = \max(X_1, u_3)$ and $Y_2 = \max(X_2, u_4)$ and write the performance as $\min(Y_1, Y_2)$, then Y_1 and Y_2 would take values u_3 and u_4 with nonzero probability. Hence the random vector $\mathbf{Y} = (Y_1, Y_2)$ would not have a density function at point (u_3, u_4), since its distribution is a mixture of continuous and discrete ones. Consequently, the push-out method would fail in its current form. To overcome this difficulty, we carry out a transformation. We first write Y_1 and Y_2 as

$$Y_1 = u_3 \max\left(\frac{X_1}{u_3}, 1\right), \quad Y_2 = u_4 \max\left(\frac{X_2}{u_4}, 1\right)$$

and then replace $\mathbf{X} = (X_1, X_2)$ by the random vector $\widetilde{\mathbf{X}} = (\widetilde{X}_1, \widetilde{X}_2)$, where

$$\widetilde{X}_1 = \max\left(\frac{X_1}{u_3}, 1\right)$$

and

$$\widetilde{X}_2 = \max\left(\frac{X_2}{u_4}, 1\right).$$

Prove that the density of the random vector $(\widetilde{X}_1, \widetilde{X}_2)$ is differentiable with respect to the variables (u_3, u_4), provided that both $\widetilde{X}_1$ and $\widetilde{X}_2$ are greater than 1.

7.11 Delta method. Let $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_m)$ be random (column) vectors, with $\mathbf{Y} = \mathbf{g}(\mathbf{X})$ for some mapping $\mathbf{g}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$. Let $\Sigma_{\mathbf{X}}$ and $\Sigma_{\mathbf{Y}}$ denote the corresponding covariance matrices. Suppose that $\mathbf{X}$ is close to its mean $\boldsymbol{\mu}$. A first-order Taylor expansion of $\mathbf{g}$ around $\boldsymbol{\mu}$ gives

$$\mathbf{Y} \approx \mathbf{g}(\boldsymbol{\mu}) + J_{\boldsymbol{\mu}}(\mathbf{g})(\mathbf{X} - \boldsymbol{\mu}),$$

where $J_{\boldsymbol{\mu}}(\mathbf{g})$ is the matrix of Jacobi of $\mathbf{g}$ (the matrix whose (i, j)-th entry is the partial derivative $\partial g_i / \partial x_j$) evaluated at $\boldsymbol{\mu}$. Show that, as a consequence,

$$\Sigma_{\mathbf{Y}} \approx J_{\boldsymbol{\mu}}(\mathbf{g}) \Sigma_{\mathbf{X}} J_{\boldsymbol{\mu}}(\mathbf{g})^\top .$$

This is called the *delta method* in statistics.

Further Reading

The SF method in the simulation context has been discovered and rediscovered independently, starting in the late 1960s. The earlier work on SF appeared in

Aleksandrov, Sysoyev, and Shemeneva [1] in 1968 and Rubinstein [15] in 1969. Motivated by the pioneering paper of Ho, Eyler, and Chien [7] on *infinitesimal perturbation analysis* (IPA) in 1979, the SF method was rediscovered at the end of the 1980s by Glynn [5] in 1990 and independently in 1989 by Reiman and Weiss [13], who called it the *likelihood ratio method*. Since then, both the IPA and SF methods have evolved and have now reached maturity; see Glasserman [4], Pflug [12], Rubinstein and Shapiro [19], and Spall [21].

To the best of our knowledge, the stochastic counterpart method in the simulation context was first suggested by Rubinstein in his PhD thesis [15]. It was applied there to estimate the optimal parameters in a complex simulation-based optimization model. It was shown numerically that the *off-line* stochastic counterpart method produces better estimates than the standard *on-line* stochastic approximation. For some later work on the stochastic counterpart method and stochastic approximation, see [16]. Alexander Shapiro should be credited for developing theoretical foundations for stochastic programs and, in particular, for the stochastic counterpart method. For relevant references, see Shapiro's elegant paper [20] and also [18, 19]. As mentioned, Geyer and Thompson [3] independently discovered the stochastic counterpart method in the early 1990s, and used it to make statistical inference in a particular unconstrained setting.

REFERENCES

1. V. M Aleksandrov, V. I. Sysoyev, and V. V. Shemeneva. Stochastic optimization. *Engineering Cybernetics*, 5:11–16, 1968.
2. S. Asmussen and P. W. Glynn. *Stochastic Simulation*. Springer-Verlag, New York, 2007.
3. C. J. Geyer and E. A. Thompson. Annealing Markov chain Monte-Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90:909–920, 1995.
4. P. Glasserman. *Gradient Estimation via Perturbation Analysis*. Kluwer, Norwell, MA, 1991.
5. P. W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.
6. D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 2nd edition, 1985.
7. Y. C. Ho, M. A. Eyler, and T. T. Chien. A gradient technique for general buffer storage design in a serial production line. *International Journal on Production Research*, 17(6):557–580, 1979.
8. J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.
9. A. J. Kleywegt and A. Shapiro. Stochastic optimization. In G. Salvendy, editor, *Handbook of Industrial Engineering*, pages 2625–2650, New York, 2001. John Wiley & Sons.
10. A. J. Kleywegt, A. Shapiro, and T. Homem de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12:479–502, 2001.

11. H. J. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag, New York, 1978.

12. G. Ch. Pflug. *Optimization of Stochastic Models*. Kluwer, Boston, 1996.

13. M. I. Reiman and A. Weiss. Sensitivity analysis for simulations via likelihood ratios. *Operations Research*, 37(5):830–844, 1989.

14. H. Robbins and S. Monro. Stochastic approximation methods. *Annals of Mathematical Statistics*, 22:400–407, 1951.

15. R. Y. Rubinstein. *Some Problems in Monte Carlo Optimization*. PhD thesis, University of Riga, Latvia, 1969. (In Russian).

16. R. Y. Rubinstein. *Monte Carlo Optimization Simulation and Sensitivity of Queueing Network*. John Wiley & Sons, New York, 1986.

17. R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling*. John Wiley & Sons, New York, 1998.

18. R. Y. Rubinstein and A. Shapiro. Optimization of static simulation models by the score function method. *Mathematics and Computers in Simulation*, 32:373–392, 1990.

19. R. Y. Rubinstein and A. Shapiro. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization via the Score Function Method*. John Wiley & Sons, New York, 1993.

20. A. Shapiro. Simulation based optimization: Convergence analysis and statistical inference. *Stochastic Models*, 12:425–454, 1996.

21. J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons, New York, 2003.

CHAPTER 8

CROSS-ENTROPY METHOD

8.1 INTRODUCTION

The *cross-entropy* (CE) method [45] is a relatively new Monte Carlo technique for both estimation and optimization. In the estimation setting, the CE method provides an adaptive way to find the optimal importance sampling distribution for quite general problems. By formulating an optimization problem as an estimation problem, the CE method becomes a general and powerful stochastic search algorithm. The method is based on a simple iterative procedure where each iteration contains two phases: (1) generate a random data sample (trajectories, vectors, etc.) according to a specified mechanism; (2) update the parameters of the random mechanism on the basis of the data in order to produce a better sample in the next iteration.

The CE method has its origins in an adaptive algorithm for rare-event estimation based on *variance minimization* (VM) [39]. This procedure was soon modified [40] to an adaptive algorithm for both rare-event estimation and combinatorial optimization, where the original VM program was replaced by a similar CE minimization program. In this chapter we present a general introduction to the CE method. For a comprehensive treatment, we refer the reader to [45].

The rest of this chapter is organized as follows: Section 8.2 presents a general CE algorithm for the estimation of rare-event probabilities, and Section 8.3 introduces a slight modification of this algorithm for solving combinatorial optimization problems. In Sections 8.4–8.6 we discuss the applications of the CE method to sev-

Simulation and the Monte Carlo Method, Third Edition. By R. Y. Rubinstein and D. P. Kroese

eral problems, such as the max-cut problem and the TSP, and provide supportive numerical results on the performance of the algorithm. In Sections 8.7 and 8.8, we show how the CE method can deal with continuous and noisy optimization problems, respectively. Last, in Section 8.9, we introduce a nonparametric version of the CE method, called the *MinxEnt* method.

8.2 ESTIMATION OF RARE-EVENT PROBABILITIES

In this section we apply the CE method in the context of efficient estimation of small probabilities. Consider, in particular, the estimation of

$$\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}}\left[I_{\{S(\mathbf{X}) \geqslant \gamma\}}\right] \tag{8.1}$$

for some fixed level γ. Here $S(\mathbf{X})$ is the sample performance, $\mathbf{X}$ is a random vector with pdf $f(\cdot;\mathbf{u})$ belonging to some parametric family $\{f(\cdot;\mathbf{v}), \mathbf{v} \in \mathscr{V}\}$, and $\{S(\mathbf{X}) \geqslant \gamma\}$ is assumed to be a rare event. We can estimate ℓ using the likelihood ratio estimator (see also (5.58))

$$\widehat{\ell} = \frac{1}{N}\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}}\, W(\mathbf{X}_k;\mathbf{u},\mathbf{v})\,, \tag{8.2}$$

where $\mathbf{X}_1,\ldots,\mathbf{X}_N$ is a random sample from $f(\mathbf{x};\mathbf{v})$ and $W(\mathbf{X}_k;\mathbf{u},\mathbf{v}) = f(\mathbf{X}_k;\mathbf{u})/f(\mathbf{X}_k;\mathbf{v})$ is the likelihood ratio.

EXAMPLE 8.1 Stochastic Shortest Path

Let us return to Example 5.15 (see also Example 5.1), where the objective is to efficiently estimate the probability ℓ that the shortest path from node A to node B in the network of Figure 8.1 has a length of at least γ. The random lengths $X_1,\ldots,X_5$ of the links are assumed to be independent and exponentially distributed with means $u_1,\ldots,u_5$, respectively.

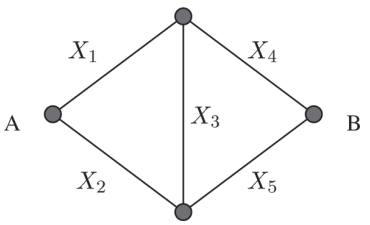

Figure 8.1: Shortest path from A to B.

We define $\mathbf{X} = (X_1,\ldots,X_5)$, $\mathbf{u} = (u_1,\ldots,u_5)$, and

$$S(\mathbf{X}) = \min\{X_1 + X_4,\ X_1 + X_3 + X_5,\ X_2 + X_5,\ X_2 + X_3 + X_4\}\,,$$

so that we can cast the problem in the framework of (8.1). As explained in Example 5.15, we can estimate (8.1) via (8.2) by drawing $X_1,\ldots,X_5$ independently from exponential distributions that are possibly *different* from the

original ones. That is, $X_i \sim \mathsf{Exp}(v_i^{-1})$ instead of $X_i \sim \mathsf{Exp}(u_i^{-1})$, $i = 1, \ldots, 5$. The corresponding likelihood ratio was given in (5.72).

The challenge is how to select a vector $\mathbf{v} = (v_1, \ldots, v_5)$ that gives the most accurate estimate of ℓ for a given simulation effort. In the toy Example 5.15, this was achieved by first choosing the trial vector $\mathbf{w}$ equal to $\mathbf{u}$ and then applying the CE updating formula (5.68), possibly iterating the latter. This approach was possible because the event $\{S(\mathbf{x}) \geqslant \gamma\}$ was *not rare*. However, for the current problem, (5.68) cannot be applied directly, since for rare events it returns, with high probability, the indeterminate expression $\frac{0}{0}$. To overcome this difficulty, we will use a different approach to selecting a good $\mathbf{v}$ by adopting a *two-stage* procedure where *both the level* γ *and the reference parameters* $\mathbf{v}$ *are updated*. One of the strengths of the CE method for rare-event simulation is that it provides a fast way to estimate accurately the optimal parameter vector $\mathbf{v}^*$.

Returning to the general situation, recall from Section 5.7 that for estimation problems of the form (8.1) the ideal (zero variance) importance sampling density is given by

$$g^*(\mathbf{x}) = \frac{f(\mathbf{x}; \mathbf{u})\, I_{\{S(\mathbf{x}) \geqslant \gamma\}}}{\ell} ,$$

which is the conditional pdf of $\mathbf{X}$ given $S(\mathbf{X}) \geqslant \gamma$. The idea behind the CE method is to get as close as possible to the optimal importance sampling distribution by using the Kullback–Leibler CE distance as a measure of closeness. Using a parametric class of densities $\{f(\mathbf{x}; \mathbf{v}), \mathbf{v} \in \mathscr{V}\}$, we see (from (5.60)) that the optimal reference parameter $\mathbf{v}^*$ is given by

$$\mathbf{v}^* = \operatorname*{argmax}_{\mathbf{v} \in \mathscr{V}} \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}} \ln f(\mathbf{X}; \mathbf{v})] . \tag{8.3}$$

We can, in principle, estimate $\mathbf{v}^*$ as

$$\operatorname*{argmax}_{\mathbf{v} \in \mathscr{V}} \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} \ln f(\mathbf{X}_k; \mathbf{v}) , \tag{8.4}$$

with $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim f(\cdot; \mathbf{u})$ — that is, using the stochastic counterpart of (8.3). However, as mentioned in Example 8.1, this is void of meaning if $\{S(\mathbf{X}) \geqslant \gamma\}$ is a rare event under $f(\cdot; \mathbf{u})$, since then most likely all indicators in the sum above will be zero.

To circumvent this problem, we will use a multilevel approach where we generate a sequence of reference parameters $\{\mathbf{v}_t,\ t \geqslant 0\}$ and a sequence of levels $\{\gamma_t,\ t \geqslant 1\}$ while iterating in both γ_t and $\mathbf{v}_t$. Our ultimate goal is to have $\mathbf{v}_t$ close to $\mathbf{v}^*$ after some number of iterations and to use $\mathbf{v}_t$ in the importance sampling density $f(\cdot; \mathbf{v}_t)$ to estimate ℓ.

We start with $\mathbf{v}_0 = \mathbf{u}$. Let ϱ be a not too small number, say $10^{-2} \leqslant \varrho \leqslant 10^{-1}$. In the first iteration, we choose $\mathbf{v}_1$ to be the optimal parameter for estimating $\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma_1)$, where γ_1 is the $(1 - \varrho)$-quantile of $S(\mathbf{X})$. That is, γ_1 is the largest real number for which

$$\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma_1) \geqslant \varrho .$$

Thus, if we simulate under $\mathbf{u}$, then level γ_1 is reached with a reasonably high probability of around ϱ. This enables us to estimate both γ_1 and $\mathbf{v}_1$ via Monte

Carlo simulation. Namely we can estimate γ_1 from a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\cdot; \mathbf{u})$ as follows: Calculate the performances $S(\mathbf{X}_i)$ for all i, and order them from smallest to largest, that is, $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. Then γ_1 is estimated via the sample $(1-\varrho)$-quantile $\widehat{\gamma}_1 = S_{(\lceil (1-\varrho)N \rceil)}$, where $\lceil a \rceil$ denotes the smallest integer larger than or equal to a (the so-called *ceiling* of a). The reference parameter $\mathbf{v}_1$ can be estimated via (8.4), replacing γ with the estimate of γ_1. Note that we can use here the *same* sample for estimating both $\mathbf{v}_1$ and γ_1. This means that $\mathbf{v}_1$ is estimated on the basis of the $\lceil \varrho N \rceil$ best samples, that is, the samples $\mathbf{X}_i$ for which $S(\mathbf{X}_i)$ is greater than or equal to $\widehat{\gamma}_1$. These form the *elite samples* in the first iteration; let N^{e} denote the number of elite samples.

In the subsequent iterations, we repeat these steps. Thus we have the following two updating phases, starting from $\mathbf{v}_0 = \widehat{\mathbf{v}}_0 = \mathbf{u}$:

1. **Adaptive updating of $\boldsymbol{\gamma_t}$.** For a fixed $\mathbf{v}_{t-1}$, let γ_t be the $(1-\varrho)$-quantile of $S(\mathbf{X})$ under $\mathbf{v}_{t-1}$. To estimate γ_t, draw a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\cdot; \widehat{\mathbf{v}}_{t-1})$ and evaluate the sample $(1-\varrho)$-quantile $\widehat{\gamma}_t$.

2. **Adaptive updating of $\mathbf{v}_t$.** For fixed γ_t and $\mathbf{v}_{t-1}$, derive $\mathbf{v}_t$ as

$$\mathbf{v}_t = \operatorname*{argmax}_{\mathbf{v} \in \mathscr{V}} \mathbb{E}_{\mathbf{v}_{t-1}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma_t\}} W(\mathbf{X}; \mathbf{u}, \mathbf{v}_{t-1}) \ln f(\mathbf{X}; \mathbf{v}) \right] . \tag{8.5}$$

   The stochastic counterpart of (8.5) is as follows: for fixed $\widehat{\gamma}_t$ and $\widehat{\mathbf{v}}_{t-1}$, derive $\widehat{\mathbf{v}}_t$ as the solution

$$\widehat{\mathbf{v}}_t = \operatorname*{argmax}_{\mathbf{v} \in \mathscr{V}} \frac{1}{N} \sum_{\mathbf{X}_k \in \mathcal{E}_t} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_{t-1}) \ln f(\mathbf{X}_k; \mathbf{v}) , \tag{8.6}$$

   where $\mathcal{E}_t$ is the *set of elite samples* in the t-th iteration, that is, the samples $\mathbf{X}_k$ for which $S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t$.

The procedure terminates when at some iteration T a level $\widehat{\gamma}_T$ is reached that is at least γ, and thus the original value of γ can be used without our getting too few samples. We then reset $\widehat{\gamma}_T$ to γ, reset the corresponding elite set, and deliver the final reference parameter $\widehat{\mathbf{v}}^*$, again using (8.6). This $\widehat{\mathbf{v}}^*$ is then used in (8.2) to estimate ℓ.

Algorithm 8.2.1: Main CE Algorithm for Rare-Event Estimation

```
Initialize v̂_0 ← u, N^e ← ⌈(1 − ϱ)N⌉, t ← 0.
Continue ← true
while Continue do
    t ← t + 1
    Generate a random sample X_1, ..., X_N from the density f(·; v̂_{t−1}).
    Calculate the performances S(X_1), ..., S(X_N).
    Order the performances from smallest to largest: S_(1) ⩽ ... ⩽ S_(N).
    Let γ̂_t ← S_(N^e).
    if γ̂_t > γ then
        γ̂_t ← γ
        Continue ← false
    Use the same sample X_1, ..., X_N to solve the stochastic program (8.6).
Generate a sample X_1, ..., X_{N_1} according to the pdf f(·; v̂_T), where T is the
  final iteration number, t, and estimate ℓ via (8.2).
```

Remark 8.2.1 In typical applications the sample size N in Line 5 can be chosen smaller than the final sample size N_1 in Line 13.

Note that Algorithm 8.2.1 breaks down the complex problem of estimating the very small probability ℓ into a sequence of simple problems, generating a sequence of pairs $\{(\widehat{\gamma}_t, \widehat{\mathbf{v}}_t)\}$, depending on ϱ, which is called the *rarity parameter*. Convergence of Algorithm 8.2.1 is discussed in [45]. Other convergence proofs for the CE method may be found in [34] and [13].

Remark 8.2.2 (Maximum Likelihood Estimator) Optimization problems of the form (8.6) appear frequently in statistics. In particular, if the W term is omitted — which will turn out to be important in CE optimization — we can recast (8.6) as

$$\widehat{\mathbf{v}}_t = \operatorname*{argmax}_{\mathbf{v}} \prod_{\mathbf{X}_k \in \mathcal{E}_t} f(\mathbf{X}_k; \mathbf{v}) \,,$$

where the product is the joint density of the elite samples. Consequently, $\widehat{\mathbf{v}}_t$ is chosen such that the joint density of the elite samples is maximized. Viewed as a function of the parameter $\mathbf{v}$, rather than of the data $\{\mathcal{E}_t\}$, this joint density is called the *likelihood*. In other words, $\widehat{\mathbf{v}}_t$ is the *maximum likelihood estimator* (it maximizes the likelihood) of $\mathbf{v}$ based on the elite samples. When the W term is present, the form of the updating formula remains similar. Recall from Section 5.7 that for exponential families the updating rules for $\widehat{\mathbf{v}}_t$ can be obtained analytically; see also Section A.3 of the Appendix.

To gain a better understanding of the CE algorithm, we also present its *deterministic* version.

Algorithm 8.2.2: Deterministic Version of the CE Algorithm

1 Initialize $\widehat{\mathbf{v}}_0 \leftarrow \mathbf{u}$, $N^{\mathrm{e}} \leftarrow \lceil (1-\varrho)N \rceil$, $t \leftarrow 0$.
2 `Continue` $\leftarrow$ **true**
3 **while** `Continue` **do**
4 $t \leftarrow t+1$
5 Calculate γ_t as

$$\gamma_t \leftarrow \max\{s \,:\, \mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geqslant s) \geqslant \varrho\} \,. \tag{8.7}$$

if $\gamma_t > \gamma$ **then**
6 $\gamma_t \leftarrow \gamma$
7 `Continue` $\leftarrow$ **false**
8 Calculate $\mathbf{v}_t$ (see (8.5)) as

$$\mathbf{v}_t \leftarrow \operatorname*{argmax}_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma_t\}} W(\mathbf{X}; \mathbf{u}, \mathbf{v}_{t-1}) \ln f(\mathbf{X}; \mathbf{v}) \right] \,. \tag{8.8}$$

Note that, when compared with Algorithm 8.2.1, Line 13 is redundant in Algorithm 8.2.2.

To provide further insight into Algorithm 8.2.1, we will apply it step by step in a number of toy examples.

EXAMPLE 8.2 Exponential Distribution

Let us revisit Examples 5.9, 5.11, and 5.13, where the goal was to estimate, via Monte Carlo simulation, the probability $\ell = \mathbb{P}_u(X \geqslant \gamma)$, with $X \sim \mathsf{Exp}(u^{-1})$.

Suppose that γ is large in comparison with u, so that $\ell = \mathrm{e}^{-\gamma/u}$ is a rare-event probability. The updating formula for $\widehat{v}_t$ in (8.6) follows from the optimization of

$$\sum_{X_k \in \mathcal{E}_t} W_k \ln\left(v^{-1}\mathrm{e}^{-X_k/v}\right) = -\sum_{X_k \in \mathcal{E}_t} W_k \ln v - \sum_{X_k \in \mathcal{E}_t} W_k \frac{X_k}{v} \,,$$

where $W_k = \mathrm{e}^{-X_k(u^{-1}-v^{-1})}v/u$. To find the maximum of the right-hand side, we take derivatives and equate the result to 0:

$$-\sum_{X_k \in \mathcal{E}_t} \frac{W_k}{v} + \sum_{X_k \in \mathcal{E}_t} \frac{W_k\, X_k}{v^2} = 0 \,.$$

Solving this for v yields $\widehat{v}_t$. Thus,

$$\widehat{v}_t = \frac{\sum_{X_k \in \mathcal{E}_t} W_k\, X_k}{\sum_{X_k \in \mathcal{E}_t} W_k} \,. \tag{8.9}$$

In other words, $\widehat{v}_t$ is simply the sample mean of the elite samples weighted by the likelihood ratios. Note that, without the weights $\{W_k\}$, we would simply have the maximum likelihood estimator of v for the $\mathsf{Exp}(v^{-1})$ distribution based on the elite samples described in Remark 8.2.2. Note further that the updating formula (8.9) follows directly from (5.68). Similarly, the *deterministic* updating formula (8.8) gives

$$v_t = \frac{\mathbb{E}_{\mathbf{u}}\left[I_{\{X \geqslant \gamma_t\}}\, X\right]}{\mathbb{E}_{\mathbf{u}}\left[I_{\{X \geqslant \gamma_t\}}\right]} = \mathbb{E}_{\mathbf{u}}[X \,|\, X \geqslant \gamma_t] = u + \gamma_t \,,$$

where γ_t is the $(1-\varrho)$-quantile of the $\mathsf{Exp}(v_{t-1}^{-1})$ distribution. Thus, $\gamma_t = -v_{t-1}\ln\varrho$.

Assume, for concreteness, that $u = 1$ and $\gamma = 32$, which corresponds to $\ell \approx 1.27 \cdot 10^{-14}$. Table 8.1 shows the evolution of $\widehat{\gamma}_t$ and $\widehat{v}_t$ for $\varrho = 0.05$ using sample size $N = 1000$. Note that iteration $t = 0$ corresponds to the original exponential pdf with expectation $u = 1$, while iterations $t = 1, 2, 3$, correspond to exponential pdfs with expectations $\widehat{v}_t$, $t = 1, 2, 3$, respectively. Figure 8.2 illustrates the iterative procedure. We see that Algorithm 8.2.1 requires three iterations to reach the final level $\widehat{\gamma}_3 = 32$. In the third iteration the lowest value of the elite samples, $S_{(N^{\mathrm{e}})}$, turns out to be greater than 32, so that in the final Line 12 of the algorithm we use $\widehat{\gamma}_3 = \gamma = 32$ instead. The corresponding reference parameter $\widehat{v}_3$ is found to be 32.82. Note that both parameters $\widehat{\gamma}_t$ and $\widehat{v}_t$ increase gradually, each time "blowing up" the tail of the exponential pdf.

The final step of Algorithm 8.2.1 now invokes the likelihood ratio estimator (8.2) to estimate ℓ, and uses a sample size N_1 that is typically larger than N.

Table 8.1: Evolution of $\widehat{\gamma}_t$ and $\widehat{v}_t$ for $\varrho = 0.05$ with $\gamma = 32$ and $N = 1000$ samples.

t	$\widehat{\gamma}_t$	$\widehat{v}_t$
0	–	1
1	2.91	3.86
2	11.47	12.46
3	32	32.82

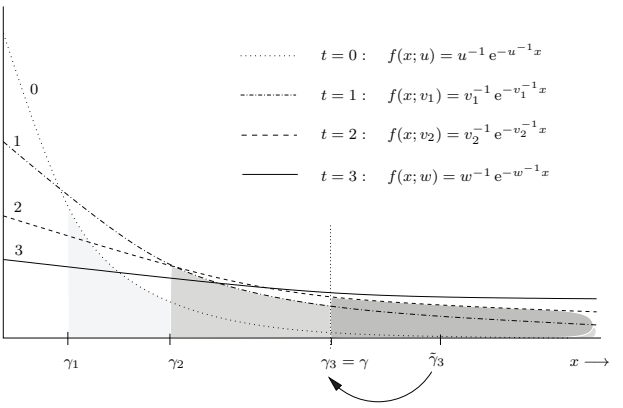

Figure 8.2: A three-level realization of Algorithm 8.2.1. Each shaded region has area ϱ.

EXAMPLE 8.3 Degeneracy

When γ is the maximum of $S(\mathbf{x})$, no "overshooting" of γ in Algorithm 8.2.1 occurs and therefore γ_t does not need to be reset. In such cases the sampling pdf may *degenerate* toward the *atomic* pdf that has all its mass concentrated at the points $\mathbf{x}$ where $S(\mathbf{x})$ is maximal. As an example, suppose we use a $\mathsf{Beta}(v, 1), v \geqslant 1$ family of importance sampling distributions, with nominal parameter $u = 1$ (corresponding to the uniform distribution), and take $S(X) = X$ and $\gamma = 1$. We find the updating formula for v from the optimization of

$$\sum_{X_k \in \mathscr{E}_k} W_k \ln(v X_k^{v-1}) = \sum_{X_k \in \mathscr{E}_k} W_k \ln v + \sum_{X_k \in \mathscr{E}_k} W_k (v-1) \ln X_k,$$

with $W_k = 1/(v X_k^{v-1})$. Hence,

$$\widehat{v}_t = \frac{\sum_{X_k \in \mathscr{E}_k} W_k}{-\sum_{X_k \in \mathscr{E}_k} W_k \ln X_k}\,.$$

Table 8.2 and Figure 8.3 show the evolution of parameters in the CE algorithm using $\varrho = 0.8$ and $N = 1000$. We see that $\widehat{\gamma}_t$ rapidly increases to γ and that the sampling pdf degenerates to the atomic density with mass at 1.

Table 8.2: The evolution of $\widehat{\gamma}_t$ and $\widehat{v}_t$ for the $\mathsf{Beta}(v, 1)$ example, with $\varrho = 0.8$ and $\gamma = 1$, using $N = 1000$ samples.

t	$\widehat{\gamma}_t$	$\widehat{v}_t$	t	$\widehat{\gamma}_t$	$\widehat{v}_t$
0	–	1	5	0.896	31.2
1	0.207	1.7	6	0.949	74.3
2	0.360	3.1	7	0.979	168.4
3	0.596	6.4	8	0.990	396.5
4	0.784	14.5	9	0.996	907.7

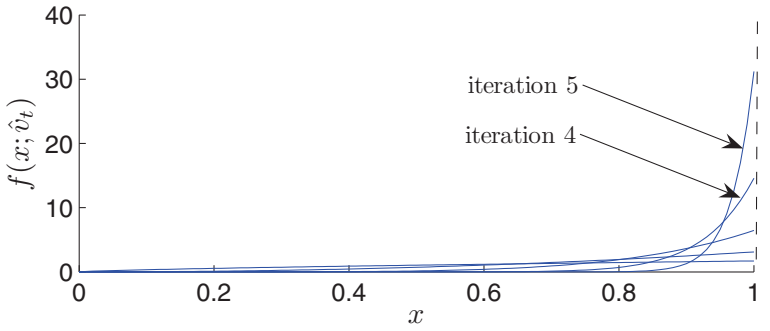

Figure 8.3: Degeneracy of the sampling distribution.

EXAMPLE 8.4 Coin Flipping

Consider the experiment where we flip n fair coins. We can describe this experiment via n independent Bernoulli random variables, $X_1, \ldots, X_n$, each with success parameter $1/2$. We write $\mathbf{X} = (X_1, \ldots, X_n) \sim \mathsf{Ber}(\mathbf{u})$, where $\mathbf{u} = (1/2, \ldots, 1/2)$ is the vector of success probabilities. Note that the range of $\mathbf{X}$ (the set of possible values it can take) contains 2^n elements. Suppose that we are interested in estimating $\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma)$, with $S(\mathbf{X}) = \sum_{k=1}^{n} X_k$. We want to employ importance sampling using $\mathbf{X} \sim \mathsf{Ber}(\mathbf{p})$ for a possibly different parameter vector $\mathbf{p} = (p_1, \ldots, p_n)$. Consider two cases: (a) $\gamma = (n+1)/2$ (with n odd) and (b) $\gamma = n$. It is readily seen that for both cases (a) and (b) the optimal importance sampling parameter vector is $\mathbf{p}^* = (1/2, \ldots, 1/2)$ and $\mathbf{p}^* = (1, \ldots, 1)$, respectively. The corresponding probabilities are $\ell = \frac{1}{2}$ and $\ell = \frac{1}{2^n}$, respectively. Note that in the first case ℓ is not a rare-event probability, but it is so for the second case (provided that n is large). Note also that in the second case $\mathsf{Ber}(\mathbf{p}^*)$ corresponds to a *degenerated* distribution that places all probability mass at the point $(1, 1, \ldots, 1)$.

Since $\{\mathsf{Ber}(\mathbf{p})\}$ forms an exponential family that is parameterized by the mean, it immediately follows from (5.68) that the updating formula for $\mathbf{p}$ in Algorithm 8.2.1 at the t-th iteration coincides with (8.9) and is given by

$$\widehat{p}_{t,i} = \frac{\sum_{\mathbf{X}_k \in \mathcal{E}_t} W_k \, X_{ki}}{\sum_{\mathbf{X}_k \mathcal{E}_t} W_k}, \quad i = 1, \ldots, n \,, \tag{8.10}$$

where X_{ki} is the i-th component of the k-th sample vector $\mathbf{X}_k \sim \mathsf{Ber}(\widehat{\mathbf{p}}_{t-1})$, and W_k is the corresponding likelihood ratio:

$$W_k = \prod_{i=1}^{n} q_i^{X_{ki}} \, r_i^{1-X_{ki}},$$

with $q_i = p_{0,i}/\widehat{p}_{t-1,i}$ and $r_i = (1 - p_{0,i})/(1 - \widehat{p}_{t-1,i})$, $i = 1, \ldots, n$. Thus, the i-th probability is updated as a weighted average of the number of 1s in the i-th position over all vectors in the elite sample.

As we will see below, this simple coin flipping example can shed light on how rare-events estimation is connected with combinatorial optimization.

Remark 8.2.3 It is important to note that if we employ the deterministic CE Algorithm 8.2.2 to any rare-event-type problem where the underlying distributions have finite supports *without fixing* γ *in advance*, it will iterate until it reaches some γ, denoted as γ_* (not necessarily the true optimal γ^*), and then stop. The corresponding importance sampling pdf $f(\mathbf{x}; \mathbf{v}_*)$ will be *degenerated*. For the coin flipping example given above, we will typically have in case (b) that $\gamma_* = \gamma^* = n$. The main Algorithm 8.2.1 behaves similarly, but in a stochastic rather than a deterministic sense. More precisely, for pdfs with finite supports and γ not fixed in advance, it will generate a tuple $(\widehat{\gamma}_T, \widehat{\mathbf{v}}_T)$ with $f(\mathbf{x}; \widehat{\mathbf{v}}_T)$ corresponding again typically to a degenerate pdf. This property of Algorithms 8.2.2 and 8.2.1 will be of crucial importance when we come to combinatorial optimization problems in the next section. As mentioned, a combinatorial optimization problem can be viewed as a rare-event estimation problem in the sense that its optimal importance sampling pdf $f(\mathbf{x}; \mathbf{v}^*)$ is degenerated and coincides with the pdf generated by the deterministic rare-event Algorithm 8.2.2, provided that it keeps iterating in γ without fixing it in advance.

In the next example, we illustrate the behavior of Algorithm 8.2.1 when applied to a typical static simulation problem of estimating $\ell = \mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$. Note that the likelihood ratio estimator $\widehat{\ell}$ of ℓ in (8.2) is of the form $\widehat{\ell} = N^{-1} \sum_{k=1}^{n} Z_k$. We measure the efficiency of the estimator by its relative error (RE), which (recall (4.6)) is defined as

$$\mathrm{RE} \;= \mathrm{Var}(\widehat{\ell})^{1/2} / \mathbb{E}[\widehat{\ell}]$$

and which is estimated by $S/(\widehat{\ell}\sqrt{N})$, with

$$S^2 = N^{-1} \sum_{k=1}^{N} Z_k^2 - (\widehat{\ell})^2$$

being the sample variance of the $\{Z_i\}$. Assuming asymptotic normality of the estimator, the confidence intervals now follow in a standard way. For example, a 95% relative confidence interval for ℓ is given by

$$\widehat{\ell} \pm 1.96\, \widehat{\ell}\, \mathrm{RE}\ .$$

■ **EXAMPLE 8.5 Stochastic Shortest Path: Example 8.1 (Continued)**

Consider again the stochastic shortest path graph of Figure 8.1. Let us take the same nominal parameter vector $\mathbf{u}$ as in Example 5.15, that is, $\mathbf{u} = (1, 1, 0.3, 0.2, 0.1)$, and estimate the probability ℓ that the minimum path length is greater than $\gamma = 6$. Note that in Example 5.15 $\gamma = 1.5$ is used, which gives rise to an event that is not rare.

Crude Monte Carlo (CMC), with 10^8 samples — a very large simulation effort — gave an estimate $8.01 \cdot 10^{-6}$ with an estimated relative error of 0.035.

To apply Algorithm 8.2.1 to this problem, we need to establish the updating rule for the reference parameter $\mathbf{v} = (v_1, \ldots, v_5)$. Since the components $X_1, \ldots, X_5$ are independent and form an exponential family parameterized by the mean, this updating formula follows immediately from (5.68), that is,

$$\widehat{v}_{t,i} = \frac{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_{t-1})\, X_{ki}}{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_{t-1})}, \quad i = 1, \ldots, 5\ , \tag{8.11}$$

with $W(\mathbf{X}; \mathbf{u}, \mathbf{v})$ given in (5.72).

We take in all our experiments with Algorithm 8.2.1 the rarity parameter $\varrho = 0.1$, the sample size in Line 5 of the algorithm $N = 10^3$, and for the final sample size $N_1 = 10^5$. Table 8.3 displays the results of Lines 1–12 of the CE algorithm. We see that after five iterations level $\gamma = 6$ is reached.

Table 8.3: Convergence of the sequence $\{(\widehat{\gamma}_t, \widehat{\mathbf{v}}_t)\}$.

t	$\widehat{\gamma}_t$	$\widehat{\mathbf{v}}_t$				
0		1.0000	1.0000	0.3000	0.2000	0.1000
1	1.1656	1.9805	2.0078	0.3256	0.2487	0.1249
2	2.1545	2.8575	3.0006	0.2554	0.2122	0.0908
3	3.1116	3.7813	4.0858	0.3017	0.1963	0.0764
4	4.6290	5.2803	5.6542	0.2510	0.1951	0.0588
5	6.0000	6.7950	6.7094	0.2882	0.1512	0.1360

Using the estimated optimal parameter vector of $\widehat{\mathbf{v}}_5 = (6.7950, 6.7094, 0.2882, 0.1512, 0.1360)$, we get in Line 13 of the CE algorithm an estimate of $7.85 \cdot 10^{-6}$, with an estimated relative error of 0.035 — the same as for the CMC method with 10^8 samples. However, whereas the CMC method required more than an hour of computation time, the CE algorithm was finished in only one second, with a Matlab implementation on a 1500 MHz computer. We see that with a minimal amount of work, we have achieved a dramatic reduction of the simulation effort.

Table 8.4 presents the performance of Algorithm 8.2.1 for the stochastic shortest path model presented above, where instead of the exponential random variables we used $\mathsf{Weib}(\alpha_i, \lambda_i)$ random variables, with $\alpha_i = 0.2$ and $\lambda_i = u_i^{-1}$, $i = 1, \ldots, 5$, where the $\{u_i\}$ are the same as before, that is, $\mathbf{u} = (1, 1, 0.3, 0.2, 0.1)$.

Table 8.4: Evolution of $\widehat{\mathbf{v}}_t$ for estimating the optimal parameter $\mathbf{v}^*$ with the TLR method and $\alpha = 0.2$. The estimated probability is $\widehat{\ell} = 3.30 \cdot 10^{-6}$, RE = 0.03.

t	$\widehat{\gamma}_t$	$\widehat{v}_{1t}$	$\widehat{v}_{2t}$	$\widehat{v}_{3t}$	$\widehat{v}_{4t}$	$\widehat{v}_{5t}$
0		1	1	1	1	1
1	3.633	2.0367	2.1279	0.9389	1.3834	1.4624
2	100.0	3.2690	3.3981	1.1454	1.3674	1.2939
3	805.3	4.8085	4.7221	0.9660	1.1143	0.9244
4	5720	6.6789	6.7252	0.6979	0.9749	1.0118
5	10000	7.5876	7.8139	1.0720	1.3152	1.2252

The Weibull distribution with shape parameter α less than 1 is an example of a *heavy-tailed* distribution. We use the TLR method (see Section 5.11) to estimate ℓ for $\gamma = 10{,}000$. Specifically, we first write (see (5.101)) $X_k = u_k Z_k^{1/\alpha}$, with $Z_k \sim \mathsf{Exp}(1)$, and then use importance sampling on the $\{Z_k\}$, changing the mean of Z_k from 1 to v_k, $k = 1, \ldots, 5$. The corresponding

updating formula is again of the form (8.11), namely,

$$\widehat{v}_{t,i} = \frac{\sum_{k=1}^{N} I_{\{\widetilde{S}(\mathbf{Z}_k) \geqslant \widehat{\gamma}_t\}} \widetilde{W}(\mathbf{Z}_k; \mathbf{1}, \widehat{\mathbf{v}}_{t-1})\, Z_{ki}}{\sum_{k=1}^{N} I_{\{\widetilde{S}(\mathbf{Z}_k) \geqslant \widehat{\gamma}_t\}} \widetilde{W}(\mathbf{Z}_k; \mathbf{1}, \widehat{\mathbf{v}}_{t-1})}, \quad i = 1, \ldots, 5\,,$$

with $\widetilde{W}(\mathbf{Z}; \mathbf{1}, \mathbf{v})$ the likelihood ratio, and $\widetilde{S}(\mathbf{Z}) = S(\mathbf{X})$. Note that the "nominal" parameter here is $\mathbf{1} = (1, 1, 1, 1, 1)$, rather than $(1, 1, 0.3, 0.2, 0.1)$. Instead of using the TLR method, one could use the standard CE method here, where the components are sampled from $\{\mathsf{Weib}(\alpha, v_i^{-1})\}$ and the $\{v_i\}$ are updated adaptively. One would obtain results similar (estimate and relative error) to those for the TLR case. The TLR is a convenient and quite general tool for importance sampling simulation, but it does not provide additional variance reduction; see also Exercises 8.5 and 8.6.

8.2.1 Root-Finding Problem

Many applications require one to estimate, for given ℓ, the *root*, γ, of the nonlinear equation

$$\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}] = \ell \tag{8.12}$$

rather than estimate ℓ itself. We call such a problem a *root-finding* problem.

An estimate of γ in (8.12) based on the sample equivalent of $\mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}]$ can be obtained, for example, via stochastic approximation; see Chapter 7 and [46]. Alternatively, we can obtain γ using the CE method. The aim is to find a good trial vector $\widehat{\mathbf{v}}_T$ such that γ can be estimated as the smallest number $\widehat{\gamma}$ such that

$$\frac{1}{N_1} \sum_{k=1}^{N_1} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_T) \leqslant \ell\,. \tag{8.13}$$

In particular our main Algorithm 8.2.1 can be modified as follows.

Algorithm 8.2.3: Root-Finding Algorithm

Initialize $\widehat{\mathbf{v}}_0 \leftarrow \mathbf{u}$, $N^{\mathrm{e}} \leftarrow \lceil (1-\varrho)N \rceil$, $t \leftarrow 0$.
`Continue` $\leftarrow$ **true**
while `Continue` **do**
  $t \leftarrow t + 1$
  Generate a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from the density $f(\cdot; \widehat{\mathbf{v}}_{t-1})$.
  Calculate the performances $S(\mathbf{X}_1), \ldots, S(\mathbf{X}_N)$.
  Order the performances from smallest to largest: $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$.
  Let $\widehat{\gamma}_t \leftarrow S_{(N^{\mathrm{e}})}$.
  Let $\widehat{\ell}_t \leftarrow \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_{t-1})$.
  if $\widehat{\ell}_t < \ell$ **then**
    $\widehat{\ell}_t \leftarrow \ell$
    `Continue` $\leftarrow$ **false**
  Determine $\widehat{\mathbf{v}}_t$ via (8.6) using the *same* sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$.
Estimate γ via (8.13) using a sample $\mathbf{X}_1, \ldots, \mathbf{X}_{N_1} \sim f(\cdot; \widehat{\mathbf{v}}_T)$, where T is the final iteration number, t.

8.2.2 Screening Method for Rare Events

Here we show how the screening method, introduced in Section 5.12, works for estimating rare-event probabilities of the form $\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}]$, where we assume, as in Section 5.12, that the components of $\mathbf{X}$ are independent, that each component is distributed according to a one-dimensional exponential family parameterized by the mean, and that $S(\mathbf{x})$ (and hence $H(\mathbf{x}) = I_{\{S(\mathbf{x}) \geqslant \gamma\}}$) is monotonically increasing in each component of $\mathbf{x}$. In particular, we will present a modification of the two-stage Algorithm 5.12.1.

As in Algorithm 5.12.1, the main idea of the first stage of our modified algorithm is to identify the bottleneck parameters *without involving the likelihood ratio*. One might wonder how this could be possible given the fact that the estimation of the rare-event probability ℓ is essentially based on likelihood ratios. The trick is to execute the first stage (the screening part) by replacing γ with some γ_0 such that $\ell_0 = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma_0)$ *is not* a rare-event probability, say $10^{-2} \leqslant \ell_0 \leqslant 10^{-1}$. As soon as γ_0 is determined, the execution of the first stage is similar to the one in Algorithm 5.12.1. It reduces to finding the estimator, say $\widehat{\mathbf{v}}_0$, of the optimal parameter vector $\mathbf{v}_0^*$ obtained from (8.4), where γ is replaced by γ_0. Note that (8.4) does not contain the likelihood ratio term $W(\mathbf{X}; \mathbf{u}, \mathbf{w})$. It is important to note again that the components of $\mathbf{v}_0^*$ are at least as large as the corresponding elements of $\mathbf{u}$, and thus we can classify the bottleneck and nonbottleneck parameters according to the relative perturbation $\delta_i = \frac{\widehat{v}_i - u_i}{u_i}$, $i = 1, \ldots, n$, which is the core of the screening algorithm.

Below we present the modified version of the two-stage screening CE-SCR Algorithm 5.12.1 suitable for rare events. We use the same notation as in Section 5.12.

Algorithm 8.2.4: Two-Stage Screening Algorithm for Rare Events

input : Sample size N, performance function S, level γ, tolerance δ, number of repetitions d.
output: Estimator $\widehat{\ell}_B$ of $\mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$.

Initialize $B \leftarrow \{1, \ldots, n\}$.
Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{u})$ and compute $\widehat{\gamma}_0$ as the $(1 - \varrho)$-sample quantile of the sample performances $\{S(\mathbf{X}_i)\}$.
for $t = 1$ **to** d **do**
    Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{u})$.
    for $i = 1$ **to** n **do**
        $\widehat{v}_i \leftarrow \dfrac{\sum_{k=1}^N I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_0\}} X_{ki}}{\sum_{k=1}^N I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_0\}}}$
        $\delta_i \leftarrow \dfrac{\widehat{v}_i - u_i}{u_i}$ `// calculate the relative perturbation`
        if $\delta_i < \delta$ **then**
            $\widehat{v}_i = u_i$
            $B \leftarrow B \setminus \{i\}$ `// remove` i `from the bottleneck set`
Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x}; \widehat{\mathbf{v}})$
$\widehat{\ell}_B \leftarrow \dfrac{1}{N} \sum_{k=1}^N I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_0\}} W_B(\mathbf{X}_{kB}; \widehat{\mathbf{v}}_B)$
return $\widehat{\ell}_B$

8.2.2.1 Numerical Results Next, we present numerical studies with Algorithm 8.2.4 for the $m \times n$ bridge system in Figure 5.7 on page 176. We are interested in estimating the rare-event probability $\ell = \mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$ that the length $S(\mathbf{X})$ of the shortest path through the graph is greater than or equal to γ, where

$$S(\mathbf{X}) = \min\{Y_{11} + \cdots + Y_{1n}, \ldots, Y_{m1} + \cdots + Y_{mn}\}$$

and the Y_{ij} are defined in (5.109). Note that the operator "max" in (5.110) is replaced by "min" here. The random variables X_{ijk} are assumed to be $\mathsf{Exp}(u_{ijk})$ distributed. As in the numerical example in Section 5.12.0.1, the $\{X_{ijk}\}$ are *not* parameterized by the mean here, so one needs to take instead $1/u_{ijk}$ and $1/\widehat{v}_{ijk}$ to compare the relative perturbations as described above. For the same reason, the parameter values corresponding to the bottleneck elements should be *smaller* than those for the nonbottleneck ones. As in Section 5.12, we purposely select (in advance) some elements of our model to be bottlenecks.

Table 8.5 presents the performance of Algorithm 8.2.4 for the 2×2 model with eight bottlenecks, using $\delta = 0.1$, $\gamma = 6$ and the sample sizes $N =$ 50,000 and $N_1 =$ 500,000. In particular, we set the bottleneck parameters $u_{111}, u_{112}, u_{121}, u_{122}, u_{211}, u_{212}, u_{221}, u_{222}$ to 1 and the remaining 12 elements to 4.

Table 8.5: Performance of Algorithm 8.2.4 for the 2×2 model. We set $\delta = 0.1$, $\gamma = 6$, $N =$ 50,000, $N_1 =$ 500,000.

	CE	VM	CE-SCR	VM-SCR
Mean $\widehat{\ell}$	2.92E-8	2.96E-8	2.88E-8	2.81E-8
Max $\widehat{\ell}$	3.93E-8	3.69E-8	3.56E-8	3.29E-8
Min $\widehat{\ell}$	2.46E-8	2.65E-8	2.54E-8	2.45E-8
RE	0.166	0.102	0.109	0.077
CPU	6.03	9.31	6.56	9.12

From the results of Table 8.5 it follows that, for this relatively small model, both CE and VM perform similarly to their screening counterparts. We will see further on that as the complexity of the model increases, VM-SCR outperforms its three alternatives and, in particular, CE-SCR.

Table 8.6 presents the typical dynamics for detecting the bottleneck parameters at the first stage of Algorithm 8.2.4 for the 2×2 model above with 20 parameters, 8 of which are bottlenecks. Similar to Table 5.3, in Table 8.6 the 0s and 1s indicate which parameters are detected as nonbottleneck and bottleneck ones, respectively, and t denotes the iteration number at the first stage of the algorithm.

Table 8.6: Typical dynamics for detecting the bottleneck parameters at the first stage of Algorithm 8.2.4.

t	u_{111}	u_{112}	u_{113}	u_{114}	u_{115}	u_{121}	u_{122}	u_{123}	u_{124}	u_{125}
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	1	0	0
2	1	1	0	0	0	1	1	0	0	0
3	1	1	0	0	0	1	1	0	0	0
4	1	1	0	0	0	1	1	0	0	0
5	1	1	0	0	0	1	1	0	0	0

t	u_{211}	u_{212}	u_{213}	u_{214}	u_{215}	u_{221}	u_{222}	u_{223}	u_{224}	u_{225}
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	0	1	1
2	1	1	1	0	0	1	1	0	1	1
3	1	1	0	0	0	1	1	0	0	0
4	1	1	0	0	0	1	1	0	0	0
5	1	1	0	0	0	1	1	0	0	0

It is readily seen that, after the first iteration, we have 13 bottleneck parameters and, after the second one, 11 bottleneck parameters; after the third iteration, the process stabilizes, delivering the 8 true bottleneck parameters.

Table 8.7 presents a typical evolution of the sequence $\{(\widehat{\gamma}_t, \widehat{\mathbf{v}}_t)\}$ for the elements of the 2×2 model given above for the VM and VM-SCR methods. We see in this table that the bottleneck elements decrease more than three times, while the nonbottleneck elements fluctuate around their nominal values 4.

Table 8.7: Typical evolution of the sequence $\{\widehat{\mathbf{v}}_t\}$ for the VM and VM-SCR methods.

	VM					VM-SCR				
t	$\widehat{v}_{111}$	$\widehat{v}_{112}$	$\widehat{v}_{113}$	$\widehat{v}_{114}$	$\widehat{v}_{115}$	$\widehat{v}_{111}$	$\widehat{v}_{112}$	$\widehat{v}_{113}$	$\widehat{v}_{114}$	$\widehat{v}_{115}$
0	1.000	1.000	4.000	4.000	4.000	1.000	1.000	4	4	4
1	0.759	0.771	3.944	3.719	3.839	0.760	0.771	4	4	4
2	0.635	0.613	3.940	3.681	3.734	0.638	0.605	4	4	4
3	0.524	0.517	4.060	3.297	3.608	0.506	0.491	4	4	4
4	0.443	0.415	3.370	3.353	3.909	0.486	0.447	4	4	4
5	0.334	0.332	3.689	3.965	4.250	0.402	0.371	4	4	4
6	0.378	0.365	3.827	3.167	4.188	0.348	0.317	4	4	4
7	0.357	0.358	3.881	4.235	4.929	0.375	0.347	4	4	4
8	0.285	0.271	4.011	2.982	4.194	0.285	0.298	4	4	4
9	0.287	0.301	3.249	2.879	3.409	0.288	0.254	4	4	4

We conclude with a larger model, consisting of a 3×10 model in which $u_{111}, u_{112}, u_{211}, u_{212}, u_{311}$, and u_{312} are chosen as bottleneck parameters and are set to 1, while the remaining parameters are set to 4. Table 8.8 presents the performance of Algorithm 8.2.4 for this model using $\delta = 0.1$, $\gamma = 6$, and $N = N_1 = 400{,}000$. In this case, both CE and VM find the true six bottlenecks. Note that VM-SCR is the most accurate of the three alternatives and that CE underestimates ℓ. Thus, for this relatively large model, CE without screening is affected by the degeneracy of the likelihood ratio, presenting a product of 150 terms.

Table 8.8: Performance of Algorithm 8.2.4 for the 3×10 model with six bottlenecks. We set $\delta = 0.1$, $\gamma = 6$, $N = 400{,}000$, $N_1 = 400{,}000$.

	CE	VM	CE-SCR	VM-SCR
Mean $\widehat{\ell}$	2.44E-8	5.34E-8	5.28E-8	5.17E-8
Max $\widehat{\ell}$	5.82E-8	7.18E-8	8.34E-8	6.93E-8
Min $\widehat{\ell}$	4.14E-15	2.76E-8	2.74E-8	4.32E-8
RE	1.05	0.28	0.33	0.15
CPU	247	482	303	531

8.2.3 CE Method Combined with Sampling from the Zero-Variance Distribution

In Algorithm 8.2.1 a general procedure is described for estimating the optimal CE parameter $\mathbf{v}^*$ using a *multilevel approach*. However, as observed in [10], such an approach may not always be necessary or desirable. We next describe how to estimate $\mathbf{v}^*$ directly from g^* without a multilevel approach. We assume that one can easily sample (approximately) from g^*. For example, we can use any of the Markov chain samplers described in Chapter 6 to simulate from g^*. Let $\mathbf{X}_1, \ldots, \mathbf{X}_N$ be approximately distributed according to g^*; then we can estimate $\mathbf{v}^*$ via $\widehat{\mathbf{v}}^* = \operatorname{argmax}_{\mathbf{v}} \sum_{k=1}^{N} \ln f(\mathbf{X}_k; \mathbf{v})$. Thus the CE program reduces to a standard maximum likelihood optimization problem. Once $\widehat{\mathbf{v}}^*$ is computed, we use the importance sampling estimator

$$\widehat{\ell} = \frac{1}{N_1} \sum_{k=1}^{N_1} \frac{f(\mathbf{X}_k; \mathbf{u})}{f(\mathbf{X}_k; \widehat{\mathbf{v}}^*)} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}}, \quad \mathbf{X}_1, \ldots, \mathbf{X}_{N_1} \sim f(\mathbf{x}; \widehat{\mathbf{v}}^*) \tag{8.14}$$

to estimate ℓ. This motivates the following single-level CE algorithm:

Algorithm 8.2.5: Single-Level CE Method for Rare-Event Estimation

input : Performance function S, level γ, sample sizes N and N_1.
output: Estimator $\widehat{\ell}$ of the rare-event probability $\ell = \mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$.

1 Run a Markov chain sampler to generate $\mathbf{X}_1, \ldots, \mathbf{X}_N$ approximately distributed according to $g^*(\mathbf{x}) \propto f(\mathbf{x})\, I_{\{S(\mathbf{x}) \geqslant \gamma\}}$.

2 Compute $\widehat{\mathbf{v}}^*$ by solving the maximum likelihood optimization problem

$$\widehat{\mathbf{v}}^* = \operatorname*{argmax}_{\mathbf{v}} \sum_{k=1}^{N} \ln f(\mathbf{X}_k; \mathbf{v}) \,.$$

3 Given $\widehat{\mathbf{v}}^*$, deliver the importance sampling estimator $\widehat{\ell}$ in (8.14).

8.3 CE METHOD FOR OPTIMIZATION

In this section we explain how the CE method works for optimization. Suppose that our task is to maximize a function $S(\mathbf{x})$ over some set $\mathscr{X}$. Let us denote the maximum by γ^*; thus

$$\gamma^* = \max_{\mathbf{x} \in \mathscr{X}} S(\mathbf{x}) \; . \tag{8.15}$$

The problem is called a *discrete* or *continuous* optimization problem based on whether $\mathscr{X}$ is discrete or continuous. An optimization problem involving both discrete and continuous variables is called a *mixed* optimization problem. A discrete optimization problem is sometimes called a *combinatorial optimization problem*, which is the main focus of this section.

The CE method takes a novel approach to optimization problems by casting the original problem (8.15) into an *estimation problem of rare-event probabilities*. By doing so, the CE method aims to locate an optimal parametric sampling distribution, that is, a probability distribution on $\mathscr{X}$, rather than locating the optimal solution directly. To this end, we define a collection of indicator functions $\{I_{\{S(\mathbf{x}) \geqslant \gamma\}}\}$ on $\mathscr{X}$ for various levels $\gamma \in \mathbb{R}$. Next, let $\{f(\cdot;\mathbf{v}), \mathbf{v} \in \mathscr{V}\}$ be a family of probability densities on $\mathscr{X}$ parameterized by a real-valued parameter vector $\mathbf{v}$. For a fixed $\mathbf{u} \in \mathscr{V}$ we associate with (8.15) the problem of estimating the rare-event probability

$$\ell(\gamma) = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma\}} \right] , \tag{8.16}$$

where $\mathbb{P}_{\mathbf{u}}$ is the probability measure under which the random state $\mathbf{X}$ has a discrete pdf $f(\cdot;\mathbf{u})$ and $\mathbb{E}_{\mathbf{u}}$ denotes the corresponding expectation operator. We call the estimation problem (8.16) the *associated stochastic problem*.

It is crucial to understand that one of the main goals of CE in optimization is to generate a sequence of pdfs $f(\cdot;\widehat{\mathbf{v}}_0), f(\cdot;\widehat{\mathbf{v}}_1), \ldots$, converging to a degenerate measure (Dirac measure) that assigns all probability mass to a single state $\mathbf{x}_T$, for which, by definition, the function value is either optimal or very close to it.

As soon as the associated stochastic problem is defined, we approximate the optimal solution, say $\mathbf{x}^*$, of (8.15) by applying Algorithm 8.2.1 for rare-event estimation, but without fixing γ in advance. It is plausible that if $\widehat{\gamma}^*$ is close to γ^*, then $f(\cdot;\widehat{\mathbf{v}}_T)$ assigns most of its probability mass close to $\mathbf{x}^*$. Thus any $\mathbf{X}$ drawn from this distribution can be used as an approximation to the optimal solution $\mathbf{x}^*$ and the corresponding function value as an approximation to the true optimal γ^* in (8.15).

To provide some more insight into the relation between combinatorial optimization and rare-event estimation, we revisit the coin flipping problem of Example 8.4, but from an optimization rather than an estimation perspective. This will serve as a preview to our discussion of *all real combinatorial optimization problems*, such as the maximal cut problem and the TSP considered in the next section. Even though the sample function $S(\mathbf{X})$ and the trajectory generation algorithm will differ from the toy example below, the updating of the sequence $\{(\gamma_t, \mathbf{v}_t)\}$ will always be determined by the *same* principles.

■ **EXAMPLE 8.6 Flipping n Coins: Example 8.4 Continued**

Suppose that we want to maximize

$$S(\mathbf{x}) = \sum_{i=1}^{n} x_i \ ,$$

where $x_i = 0$ or 1 for all $i = 1, \ldots, n$. Clearly, the optimal solution to (8.15) is $\mathbf{x}^* = (1, \ldots, 1)$. The simplest way to put the deterministic program (8.15) into a stochastic framework is to associate with each component $x_i,\ i = 1, \ldots, n$ a Bernoulli random variable $X_i,\ i = 1, \ldots, n$. For simplicity, we can assume that all $\{X_i\}$ are independent and that each component i has success probability $1/2$. By doing so, we turn the associated stochastic problem (8.16) into a rare-event estimation problem. Taking into account that there is a single solution $\mathbf{x}^* = (1, \ldots, 1)$, using the CMC method, we obtain $\ell(\gamma^*) = 1/|\mathscr{X}|$, where $|\mathscr{X}| = 2^n$, which for large n is a very small probability. Instead of estimating $\ell(\gamma)$ via CMC, we can estimate it via importance sampling using $X_i \sim \mathsf{Ber}(p_i)$, $i = 1, \ldots, n$.

The next step is, clearly, to apply Algorithm 8.2.1 to (8.16) without fixing γ in advance. As mentioned in Remark 8.2.3, CE Algorithm 8.2.1 should be viewed as the stochastic counterpart of the deterministic CE Algorithm 8.2.2, and the latter will iterate until it reaches a local maximum. We thus obtain a sequence $\{\widehat{\gamma}_t\}$ that converges to a local or global maximum, which can be taken as an estimate for the true optimal solution γ^*.

In summary, in order to solve a combinatorial optimization problem, we will employ the CE Algorithm 8.2.1 for rare-event estimation without fixing γ in advance. By doing so, we can treat the CE algorithm for optimization as a modified version of Algorithm 8.2.1. In particular, by analogy to Algorithm 8.2.1, we choose a not very small number ϱ, say $\varrho = 10^{-2}$, initialize the parameter vector $\mathbf{u}$ by setting $\mathbf{v}_0 = \mathbf{u}$, and proceed as follows:

1. **Adaptive updating of $\boldsymbol{\gamma_t}$.** For a fixed $\mathbf{v}_{t-1}$, let γ_t be the $(1-\varrho)$-quantile of $S(\mathbf{X})$ under $\mathbf{v}_{t-1}$. As before, an estimator $\widehat{\gamma}_t$ of γ_t can be obtained by drawing a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\cdot; \mathbf{v}_{t-1})$ and then evaluating the sample $(1-\varrho)$-quantile of the performances as

$$\widehat{\gamma}_t = S_{(\lceil (1-\varrho)N \rceil)} \ . \tag{8.17}$$

2. **Adaptive updating of $\mathbf{v}_t$.** For fixed γ_t and $\mathbf{v}_{t-1}$, obtain $\mathbf{v}_t$ as the solution of the program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}}[I_{\{S(\mathbf{X}) \geqslant \gamma_t\}} \ln f(\mathbf{X}; \mathbf{v})] \ . \tag{8.18}$$

The stochastic counterpart of (8.18) is then as follows: for fixed $\widehat{\gamma}_t$ and $\widehat{\mathbf{v}}_{t-1}$, obtain $\widehat{\mathbf{v}}_t$ as the solution of the following program:

$$\max_{\mathbf{v}} \widehat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} \ln f(\mathbf{X}_k; \mathbf{v}) \ . \tag{8.19}$$

It is important to observe that, in contrast to (8.5) and (8.6) (for the rare-event setting), (8.18) and (8.19) *do not contain the likelihood ratio terms* W. The reason is that in the rare-event setting the initial (nominal) parameter $\mathbf{u}$ is specified in advance and is an essential part of the estimation problem. In contrast, the initial reference vector $\mathbf{u}$ in the associated stochastic problem is quite arbitrary. In effect, by dropping the W term, we can efficiently estimate at each iteration t the CE optimal reference parameter vector $\mathbf{v}_t$ for the rare-event probability $\mathbb{P}_{\mathbf{v}_t}(S(\mathbf{X}) \geqslant \gamma_t) \geq \mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geqslant \gamma_t)$, even for high-dimensional problems.

Remark 8.3.1 (Smoothed Updating) Instead of updating the parameter vector $\mathbf{v}$ directly via the solution of (8.19), we use the *smoothed* version

$$\widehat{\mathbf{v}}_t = \alpha \widetilde{\mathbf{v}}_t + (1-\alpha)\widehat{\mathbf{v}}_{t-1}, \tag{8.20}$$

where $\widetilde{\mathbf{v}}_t$ is the parameter vector obtained from the solution of (8.19) and α is called the *smoothing parameter*, and typically $0.7 < \alpha \leqslant 1$. Clearly, for $\alpha = 1$ we have our original updating rule. The reason for using the smoothed (8.20) instead of the original updating rule is twofold: (a) to smooth out the values of $\widehat{\mathbf{v}}_t$ and (b) to reduce the probability that some component $\widehat{v}_{t,i}$ of $\widehat{\mathbf{v}}_t$ will be 0 or 1 at the first few iterations. This is particularly important when $\widehat{\mathbf{v}}_t$ is a vector or matrix of *probabilities*. Note that for $0 < \alpha \leqslant 1$, we always have $\widehat{v}_{t,i} > 0$, while for $\alpha = 1$, we might have (even at the first iterations) $\widehat{v}_{t,i} = 0$ or $\widehat{v}_{t,i} = 1$ for some indexes i. As result, the algorithm will converge to a wrong solution.

Thus, the main CE optimization algorithm, which includes smoothed updating of parameter vector $\mathbf{v}$ and which presents a slight modification of Algorithm 8.2.1 can be summarized as follows.

Algorithm 8.3.1: Main CE Algorithm for Optimization

1 Initialize $\widehat{\mathbf{v}}_0 \leftarrow \widehat{\mathbf{v}}_0$.
2 **repeat**
3 $\quad t \leftarrow t+1$
4 $\quad$ Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from the density $f(\cdot; \widehat{\mathbf{v}}_{t-1})$ and compute the sample $(1-\varrho)$-quantile $\widehat{\gamma}_t$ of the performances according to (8.17).
5 $\quad$ Use the *same* sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ and solve the stochastic program (8.19). Denote the solution by $\widetilde{\mathbf{v}}_t$.
6 $\quad$ Apply (8.20) to smooth out the vector $\widetilde{\mathbf{v}}_t$.
7 **until** *a stopping criterion is met.*

Remark 8.3.2 (Minimization) When $S(\mathbf{x})$ is to be *minimized* instead of maximized, we simply change the inequalities "$\geqslant$" to "$\leqslant$" and take the ϱ-quantile instead of the $(1-\varrho)$-quantile. Alternatively, we can just maximize $-S(\mathbf{x})$.

As a stopping criterion, we can use, for example: if for some $t \geqslant d$, say $d = 5$,

$$\widehat{\gamma}_t = \widehat{\gamma}_{t-1} = \cdots = \widehat{\gamma}_{t-d}\ , \tag{8.21}$$

then stop. As an alternative estimate for γ^*, we can consider

$$\widetilde{\gamma}_T = \max_{0 \leqslant s \leqslant T} \widehat{\gamma}_s\ . \tag{8.22}$$

Note that the initial vector $\widehat{\mathbf{v}}_0$, the sample size N, the stopping parameter d, and the number ϱ have to be specified in advance, but the rest of the algorithm is "self-tuning". Note also that, by analogy to the simulated annealing algorithm, γ_t may be viewed as the "annealing temperature". In contrast to simulated annealing, where the cooling scheme is chosen in advance, in the CE algorithm it is updated adaptively.

EXAMPLE 8.7 Flipping Coins: Example 8.6 (Continued)

In this case, the random vector $\mathbf{X} = (X_1, \ldots, X_n) \sim \mathsf{Ber}(\mathbf{p})$ and the parameter vector $\mathbf{v}$ is $\mathbf{p}$. Consequently, the pdf is

$$f(\mathbf{X}; \mathbf{p}) = \prod_{i=1}^{n} p_i^{X_i} (1 - p_i)^{1 - X_i} ,$$

and since each X_i can only be 0 or 1,

$$\frac{\partial}{\partial p_i} \ln f(\mathbf{X}; \mathbf{p}) = \frac{X_i}{p_i} - \frac{1 - X_i}{1 - p_i} = \frac{1}{(1 - p_i) p_i} (X_i - p_i) .$$

Now we can find the optimal parameter vector $\mathbf{p}$ of (8.19) by setting the first derivatives with respect to p_i equal to zero for $i = 1, \ldots, n$, that is,

$$\frac{\partial}{\partial p_i} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} \ln f(\mathbf{X}_k; \mathbf{p}) = \frac{1}{(1 - p_i)\, p_i} \sum_{i=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} (X_{ki} - p_i) = 0 .$$

Thus, we obtain

$$p_i = \frac{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} X_{ki}}{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}}} , \tag{8.23}$$

which gives the same updating formula as (8.10) *except for the W term*. Recall that the updating formula (8.23) holds for all one-dimensional exponential families that are parameterized by the mean; see (5.68). Note also that the parameters are simply updated via their maximum likelihood estimators, using only the elite samples; see Remark 8.2.2.

Algorithm 8.3.1 can, in principle, be applied to any discrete and continuous optimization problem. However, for each problem two essential actions need to be taken:

1. We need to specify how the samples are generated. In other words, we need to specify the family of densities $\{f(\cdot; \mathbf{v})\}$.

2. We need to update the parameter vector $\mathbf{v}$ based on CE minimization program (8.19), which is the *same* for all optimization problems.

In general, there are many ways to generate samples from $\mathscr{X}$, and it is not always immediately clear which method will yield better results or easier updating formulas.

Remark 8.3.3 (Parameter Selection) The choice of the sample size N and the rarity parameter ϱ depends on the size of the problem and the number of parameters in the associated stochastic problem. Typical choices are $\varrho = 0.1$ or $\varrho = 0.01$, and $N = c\,K$, where K is the number of parameters that need to be estimated/updated and c is a constant between 1 and 10.

By analogy to Algorithm 8.2.2, we also present the deterministic version of Algorithm 8.3.1, which will be used below.

Algorithm 8.3.2: Deterministic CE Algorithm for Optimization

1 Choose some $\mathbf{v}_0$.
2 **repeat**
3 Calculate γ_t as

$$\gamma_t \leftarrow \max\left\{s \,:\, \mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geqslant s) \geqslant \varrho\right\}. \tag{8.24}$$

4 Calculate $\mathbf{v}_t$ as

$$\mathbf{v}_t \leftarrow \operatorname*{argmax}_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}}\left[I_{\{S(\mathbf{X})\geqslant\gamma_t\}} \ln f(\mathbf{X};\mathbf{v})\right]. \tag{8.25}$$

5 **until** $\gamma_t = \gamma_{t-1} = \cdots = \gamma_{t-d}$ `// for, say,` $d = 5$

Remark 8.3.4 Note that instead of the CE distance we could minimize the variance of the estimator, as discussed in Section 5.7. As mentioned, the main reason for using CE is that for exponential families the parameters can be updated analytically, rather than numerically as for the VM procedure.

Below we present several applications of the CE method to combinatorial optimization, namely the max-cut, the bipartition, and the TSP. We demonstrate numerically the efficiency of the CE method and its fast convergence for several case studies. For additional applications of CE, see [45] and the list of references at the end of this chapter.

8.4 MAX-CUT PROBLEM

The maximal cut or *max-cut* problem can be formulated as follows: Given a graph $G = G(V, E)$ with a set of nodes $V = \{1, \ldots, n\}$ and a set of edges E between the nodes, partition the nodes of the graph into two arbitrary subsets V_1 and V_2 such that the sum of the weights (costs) c_{ij} of the edges going from one subset to the other is maximized. Note that some of the c_{ij} may be 0 — indicating that there is actually no edge from i to j.

As an example, consider the graph in Figure 8.4, with corresponding cost matrix $C = (c_{ij})$ given by

$$C = \begin{pmatrix} 0 & 2 & 2 & 5 & 0 \\ 2 & 0 & 1 & 0 & 3 \\ 2 & 1 & 0 & 4 & 2 \\ 5 & 0 & 4 & 0 & 1 \\ 0 & 3 & 2 & 1 & 0 \end{pmatrix}. \tag{8.26}$$

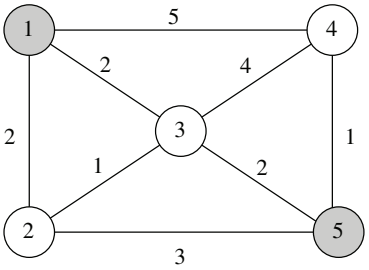

Figure 8.4: A six-node network with the cut $\{\{1,5\},\{2,3,4\}\}$.

Here the cut $\{\{1,5\},\{2,3,4\}\}$ has cost

$$c_{12}+c_{13}+c_{14}+c_{52}+c_{53}+c_{54}=2+2+5+3+2+1=15\ .$$

A cut can be conveniently represented via its corresponding *cut vector* $\mathbf{x}=(x_1,\ldots,x_n)$, where $x_i=1$ if node i belongs to same partition as 1 and 0 otherwise. For example, the cut in Figure 8.4 can be represented via the cut vector $(1,0,0,0,1)$. For each cut vector $\mathbf{x}$, let $\{V_1(\mathbf{x}),V_2(\mathbf{x})\}$ be the partition of V induced by $\mathbf{x}$, such that $V_1(\mathbf{x})$ contains the set of indexes $\{i:x_i=1\}$. If not stated otherwise, we set $x_1=1\in V_1$.

Let $\mathscr{X}$ be the set of all cut vectors $\mathbf{x}=(1,x_2,\ldots,x_n)$, and let $S(\mathbf{x})$ be the corresponding cost of the cut. Then

$$S(\mathbf{x})=\sum_{i\in V_1(\mathbf{x}),\,j\in V_2(\mathbf{x})} c_{ij}\ . \tag{8.27}$$

It is readily seen that the total number of cut vectors is

$$|\mathscr{X}|=2^{n-1}. \tag{8.28}$$

We will assume below that the graph is *undirected*. Note that for a *directed* graph the cost of a cut $\{V_1,V_2\}$ includes the cost of the edges both from V_1 to V_2 and from V_2 to V_1. In this case, the cost corresponding to a cut vector $\mathbf{x}$ is therefore

$$S(\mathbf{x})=\sum_{i\in V_1(\mathbf{x}),\,j\in V_2(\mathbf{x})} (c_{ij}+c_{ji})\ . \tag{8.29}$$

Next, we generate random cuts and update of the corresponding parameters using the CE Algorithm 8.3.1. The most natural and easiest way to generate the cut vectors is to let $X_2,\ldots,X_n$ be independent Bernoulli random variables with success probabilities $p_2,\ldots,p_n$.

Algorithm 8.4.1: Random Cuts Generation

1 Generate an n-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_n)$ from $\mathsf{Ber}(\mathbf{p})$ with independent components, where $\mathbf{p} = (1, p_2, \ldots, p_n)$.
2 Construct the partition $\{V_1(\mathbf{X}), V_2(\mathbf{X})\}$ of V and calculate the performance $S(\mathbf{X})$ as in (8.27).

The updating formulas for $\widehat{p}_{t,i}$ are the same as for the toy Example 8.7 and are given in (8.23).

The following toy example illustrates, step by step, the workings of the deterministic CE Algorithm 8.3.2. The small size of the problem allows us to make all calculations analytically, that is, using directly the updating rules (8.24) and (8.25) rather than their stochastic counterparts.

■ **EXAMPLE 8.8 Illustration of Algorithm 8.3.2**

Consider the five-node graph presented in Figure 8.4. The 16 possible cut vectors (see (8.28)) and the corresponding cut values are given in Table 8.9.

Table 8.9: Possible cut vectors of Example 8.8.

$\mathbf{X}$	V_1	V_2	$S(\mathbf{X})$
(1,0,0,0,0)	$\{1\}$	$\{2,3,4,5\}$	9
(1,1,0,0,0)	$\{1,2\}$	$\{3,4,5\}$	11
(1,0,1,0,0)	$\{1,3\}$	$\{2,4,5\}$	14
(1,0,0,1,0)	$\{1,4\}$	$\{2,3,5\}$	9
(1,0,0,0,1)	$\{1,5\}$	$\{2,3,4\}$	15
(1,1,1,0,0)	$\{1,2,3\}$	$\{4,5\}$	14
(1,1,0,1,0)	$\{1,2,4\}$	$\{3,5\}$	11
(1,1,0,0,1)	$\{1,2,5\}$	$\{3,4\}$	11
(1,0,1,1,0)	$\{1,3,4\}$	$\{2,5\}$	6
(1,0,1,0,1)	$\{1,3,5\}$	$\{2,4\}$	16
(1,0,0,1,1)	$\{1,4,5\}$	$\{2,3\}$	13
(1,1,1,1,0)	$\{1,2,3,4\}$	$\{5\}$	6
(1,1,1,0,1)	$\{1,2,3,5\}$	$\{4\}$	10
(1,1,0,1,1)	$\{1,2,4,5\}$	$\{3\}$	9
(1,0,1,1,1)	$\{1,3,4,5\}$	$\{2\}$	6
(1,1,1,1,1)	$\{1,2,3,4,5\}$	$\emptyset$	0

Clearly, in this case the optimal cut vector is $\mathbf{x}^* = (1,0,1,0,1)$ with $S(\mathbf{x}^*) = \gamma^* = 16$.

We will show next that in the deterministic Algorithm 8.3.2, adapted to the max-cut problem, the parameter vectors $\mathbf{p}_0, \mathbf{p}_1, \ldots$ converge to the optimal $\mathbf{p}^* = (1,0,1,0,1)$ after two iterations, provided that $\varrho = 10^{-1}$ and $\mathbf{p}_0 = (1, 1/2, 1/2, 1/2, 1/2)$.

Iteration 1

In the first step of the first iteration, we have to determine γ_1 from

$$\gamma_t = \max \{\gamma \text{ s.t. } \mathbb{E}_{\mathbf{p}_{t-1}}[I_{\{S(\mathbf{X})\geqslant\gamma\}}] \geqslant 0.1\} . \tag{8.30}$$

It is readily seen that under the parameter vector $\mathbf{p}_0$, $S(\mathbf{X})$ takes values in $\{0, 6, 9, 10, 11, 13, 14, 15, 16\}$ with probabilities $\{1/16, 3/16, 3/16, 1/16, 3/16, 1/16, 2/16, 1/16, 1/16\}$. Hence we find $\gamma_1 = 15$. In the second step, we need to solve

$$\mathbf{p}_t = \underset{\mathbf{p}}{\operatorname{argmax}} \, \mathbb{E}_{\mathbf{p}_{t-1}} \left[I_{\{S(\mathbf{X})\geqslant\gamma_t\}} \ln f(\mathbf{X}; \mathbf{p}) \right] , \tag{8.31}$$

which has the solution

$$p_{t,i} = \frac{\mathbb{E}_{\mathbf{p}_{t-1}}[I_{\{S(\mathbf{X})\geqslant\gamma_t\}} X_i]}{\mathbb{E}_{\mathbf{p}_{t-1}}[I_{\{S(\mathbf{X})\geqslant\gamma_t\}}]} .$$

There are only two vectors $\mathbf{x}$ for which $S(\mathbf{x}) \geqslant 15$, namely, $(1, 0, 0, 0, 1)$ and $(1, 0, 1, 0, 1)$, and both have probability $1/16$ under $\mathbf{p}_0$. Thus,

$$p_{1,i} = \begin{cases} \dfrac{2/16}{2/16} = 1 & \text{for } i = 1, 5, \\ \dfrac{1/16}{2/16} = \dfrac{1}{2} & \text{for } i = 3, \\ \dfrac{0}{2/16} = 0 & \text{for } i = 4, 2. \end{cases}$$

Iteration 2

In the second iteration $S(\mathbf{X})$ is 15 or 16 with probability $1/2$. Applying again (8.30) and (8.31) yields the optimal $\gamma_2 = 16$ and the optimal $\mathbf{p}_2 = (1, 0, 1, 0, 1)$, respectively.

Remark 8.4.1 (Alternative Stopping Rule) Note that the stopping rule (8.21), which is based on convergence of the sequence $\{\widehat{\gamma}_t\}$ to γ^*, stops Algorithm 8.3.1 when the sequence $\{\widehat{\gamma}_t\}$ does not change. An alternative stopping rule is to stop when the sequence $\{\widehat{\mathbf{p}}_t\}$ is very close to a degenerated one, for example if $\min\{\widehat{p}_i, 1 - \widehat{p}_i\} < \varepsilon$ for all i, where ε is some small number.

The code in Table 8.5 gives a simple Matlab implementation of the CE algorithm for the max-cut problem, with cost matrix (8.26). Note that, although the max-cut examples presented here are of relatively small size, basically the *same* CE program can be used to tackle max-cut problems of much higher dimension, comprising hundreds or thousands of nodes.

```
global C;
C = [ 0  2  2  5  0;                        % cost matrix
      2  0  1  0  3;
      2  1  0  4  2;
      5  0  4  0  1;
      0  3  2  1  0];
m = 5; N = 100; Ne = 10; eps = 10^-3; p = 1/2*ones(1,m); p(1) = 1;
while max(min(p,1-p)) > eps
   x = (rand(N,m) < ones(N,1)*p);           % generate cut vectors
   SX = S(x);
   sortSX = sortrows([x SX], m+1);
   p = mean(sortSX(N-Ne+1:N, 1:m))          % update the parameters
end

function perf = S(x)                        % performance function
global C;
N = size(x,1);
for i=1:N
   V1 = find(x(i,:));                       % {V1,V2} is the partition
   V2 = find(~x(i,:));
perf(i,1) = sum(sum(C(V1,V2)));             % size of the cut
end
```

Figure 8.5: Matlab CE program to solve the max-cut problem with cost matrix (8.26).

EXAMPLE 8.9 Maximal Cuts for the Dodecahedron Graph

To further illustrate the behavior of the CE algorithm for the max-cut problem, consider the so-called *dodecahedron graph* in Figure 8.6. Suppose that all edges have cost 1. We wish to partition the node set into two subsets (color the nodes black and white) such that the cost across the cut, given by (8.27), is maximized. Although this problem exhibits a lot of symmetry, it is not clear beforehand what the solution(s) should be.

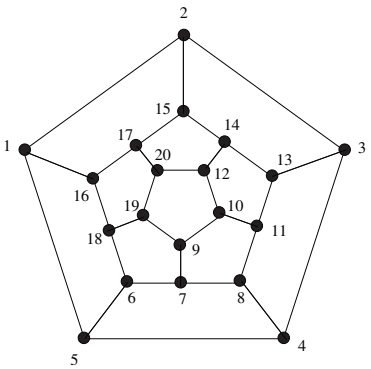

Figure 8.6: Dodecahedron graph.

The performance of the CE algorithm is depicted in Figure 8.7 using $N = 200$ and $\varrho = 0.1$.

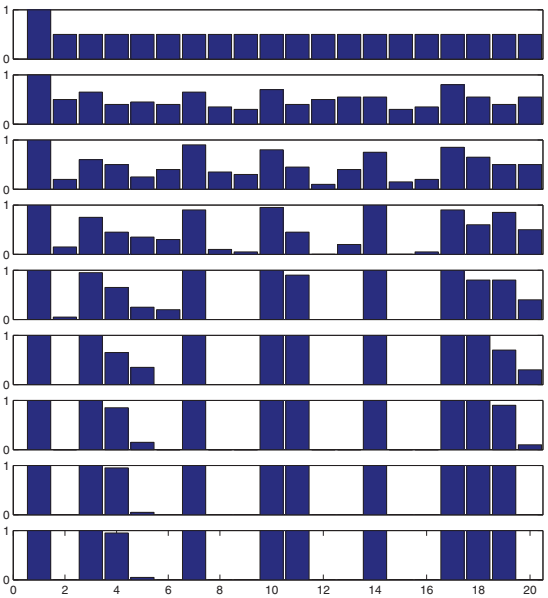

Figure 8.7: Evolution of the CE algorithm for the dodecahedron max-cut problem.

Observe that the probability vector $\widehat{\mathbf{p}}_t$ quickly (eight iterations) converges to a degenerate vector — corresponding (for this particular case) to the solution $\mathbf{x}^* = (1,0,1,1,0,0,1,0,0,1,1,0,\ 0,1,0,0,1,1,1,0)$. Thus, $V_1^* = \{1,3,4,7,10,11,14,17,18,19\}$. This required around 1600 function evaluations, as compared to $2^{19}-1 \approx 5\cdot 10^5$ if all cut vectors were to be enumerated. The maximal value is 24. It is interesting that, because of the symmetry, there are in fact many optimal solutions. We found that during each run the CE algorithm "focuses" on one (not always the same) of the solutions.

Max-Cut Problem with r Partitions

We can readily extend the max-cut procedure to the case where the node set V is partitioned into $r > 2$ subsets $\{V_1, \ldots, V_r\}$ such that the sum of the total weights of all edges going from subset V_a to subset V_b, $a, b = 1, \ldots, r$, $(a < b)$ is maximized. Thus, for each partition $\{V_1, \ldots, V_r\}$, the value of the objective function is

$$\sum_{a=1}^{r} \sum_{b=a+1}^{r} \sum_{i \in V_a,\, j \in V_b} c_{ij} \ .$$

In this case, we can follow the basic steps of Algorithm 8.3.1 using independent r-point distributions, instead of independent Bernoulli distributions, and update the probabilities as

$$\widehat{p}_{t,ij} = \frac{\sum_{\mathbf{X}_k \in \mathscr{E}_t} I_{\{X_{ki}=j\}}}{|\mathscr{E}_t|} \ . \tag{8.32}$$

8.5 PARTITION PROBLEM

The partition problem is similar to the max-cut problem. The only difference is that the *size* of each class is *fixed* in advance. This has implications for the trajectory generation. Consider, for example, a partition problem in which V has to be partitioned into two *equal* sets, assuming that n is even. We could simply use Algorithm 8.4.1 for the random cut generation, that is, generate $\mathbf{X} \sim \mathsf{Ber}(\mathbf{p})$ and *reject* partitions that have unequal size, but this would be highly inefficient. We can speed up this method by drawing directly from the *conditional* distribution of $\mathbf{X} \sim \mathsf{Ber}(\mathbf{p})$ given $X_1 + \cdots + X_n = n/2$. The parameter $\mathbf{p}$ is then updated in exactly the same way as before. Unfortunately, generating from a conditional Bernoulli distribution is not as straightforward as generating independent Bernoulli random variables. A useful technique is the so-called *drafting* method. We provide computer code for this method in Section A.2 of the Appendix.

As an alternative, we describe next a simple algorithm for the generation of a random bipartition $\{V_1, V_2\}$ with exactly m elements in V_1 and $n - m$ elements in V_2 that works well in practice. Extension of the algorithm to r-partition generation is simple.

The algorithm requires the generation of random permutations $\Pi = (\Pi_1, \ldots, \Pi_n)$ of $(1, \ldots, n)$, uniformly over the space of all permutations. This can be done via Algorithm 2.10.2. We demonstrate our algorithm first for a five-node network, assuming $m = 2$ and $m - n = 3$ for a given vector $\mathbf{p} = (p_1, \ldots, p_5)$.

■ **EXAMPLE 8.10 Generating a Bi-Partition for $m = 2$ and $n = 5$**

1. Generate a random permutation $\Pi = (\Pi_1, \ldots, \Pi_5)$ of $(1, \ldots, 5)$, uniformly over the space of all 5! permutations. Let $(\pi_1, \ldots, \pi_5)$ be a particular outcome, for example, $(\pi_1, \ldots, \pi_5) = (3, 5, 1, 2, 4)$. This means that we will draw independent Bernoulli random variables in the following order: $\mathsf{Ber}(p_3)$, $\mathsf{Ber}(p_5)$, $\mathsf{Ber}(p_1), \ldots$.

2. Given $\Pi = (\pi_1, \ldots, \pi_5)$ and the vector $\mathbf{p} = (p_1, \ldots, p_5)$, generate independent Bernoulli random variables $X_{\pi_1}, X_{\pi_2}, \ldots$ from $\mathsf{Ber}(p_{\pi_1})$, $\mathsf{Ber}(p_{\pi_2})$, $\ldots$, respectively, *until either exactly* $m = 2$ unities or $n - m = 3$ zeros are generated. Note that, in general, the number of samples is a random variable with the range from $\min\{m, n - m\}$ to n. Assume, for concreteness, that the first four independent Bernoulli samples (from $\mathsf{Ber}(p_3), \mathsf{Ber}(p_5), \mathsf{Ber}(p_1), \mathsf{Ber}(p_2)$ given above) result in the following outcome $(0, 0, 1, 0)$. Since we have already generated three 0s, we can set $X_4 \equiv 1$ and deliver $\{V_1(\mathbf{X}), V_2(\mathbf{X})\} = \{(1, 4), (2, 3, 5)\}$ as the desired partition.

3. If in the previous step $m = 2$ unities are generated, set the remaining three elements to 0; if instead three 0s are generated, set the remaining two elements to 1 and deliver $\mathbf{X} = (X_1, \ldots, X_n)$ as the final partition vector. Construct the partition $\{V_1(\mathbf{X}), V_2(\mathbf{X})\}$ of V.

With this example in hand, the random partition generation algorithm can be written as follows:

Algorithm 8.5.1: Random Partition Generation Algorithm

1 Generate a random permutation $\Pi = (\Pi_1, \ldots, \Pi_n)$ of $(1, \ldots, n)$ uniformly over the space of all $n!$ permutations.
2 Given $\Pi = (\pi_1, \ldots, \pi_n)$, independently generate Bernoulli random variables $X_{\pi_1}, X_{\pi_2}, \ldots$ from $\mathsf{Ber}(p_{\pi_1}), \mathsf{Ber}(p_{\pi_2}), \ldots$, respectively, *until* m 1s or $n-m$ 0s are generated.
3 If in the previous step m 1s are generated, set the remaining elements to 0; if, on the other hand, $n-m$ 0s are generated, set the remaining elements to 1s. Deliver $\mathbf{X} = (X_1, \ldots, X_n)$ as the final partition vector.
4 Construct the partition $\{V_1(\mathbf{X}), V_2(\mathbf{X})\}$ of V and calculate the performance $S(\mathbf{X})$ according to (8.27).

We take the updating formula for the reference vector $\mathbf{p}$ *exactly the same* as in (8.10).

8.5.1 Empirical Computational Complexity

Finally, we come to consider the computational complexity of Algorithm 8.3.1 for the max-cut and the partition problems, which can be defined as

$$\kappa_n = T_n(N_n G_n + U_n) \,. \tag{8.33}$$

Here T_n is the total number of iterations needed before Algorithm 8.3.1 stops; N_n is the sample size, that is, the total number of maximal cuts and partitions generated at each iteration; G_n is the cost of generating the random Bernoulli vectors of size n for Algorithm 8.3.1; $U_n = \mathcal{O}(N_n n^2)$ is the cost of updating the tuple $(\widehat{\gamma}_t, \widehat{\mathbf{p}}_t)$. The last follows from the fact that computing $S(\mathbf{X})$ in (8.27) is a $\mathcal{O}(n^2)$ operation.

For the model in (8.62) we found empirically that $T_n = \mathcal{O}(\ln n)$, provided that $100 \leqslant n \leqslant 1000$. For the max-cut problem, considering that we take $n \leqslant N_n \leqslant 10\, n$ and that G_n is $\mathcal{O}(n)$, we obtain $\kappa_n = \mathcal{O}(n^3 \ln n)$. In our experiments, the complexity we observed was more like

$$\kappa_n = \mathcal{O}(n \ln n) \,.$$

The partition problem has similar computational characteristics. It is important to note that these empirical complexity results are solely for the model with the cost matrix (8.62).

8.6 TRAVELING SALESMAN PROBLEM

The CE method can also be applied to solve the traveling salesman problem (TSP). Recall (see Example 6.12 for a more detailed formulation) that the objective is to find the shortest tour through all the nodes in a graph G. As in Example 6.12, we assume that the graph is complete and that each tour is represented as a permutation $\mathbf{x} = (x_1, \ldots, x_n)$ of $(1, \ldots, n)$. Without loss of generality, we can set $x_1 = 1$, so that the set of all possible tours $\mathscr{X}$ has cardinality $|\mathscr{X}| = (n-1)!$. Let $S(\mathbf{x})$ be the total length of tour $\mathbf{x} \in \mathscr{X}$, and let $C = (c_{ij})$ be the cost matrix. Our goal is thus to solve

$$\min_{\mathbf{x} \in \mathscr{X}} S(\mathbf{x}) \quad = \quad \min_{\mathbf{x} \in \mathscr{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i, x_{i+1}} + c_{x_n, 1} \right\} \,. \tag{8.34}$$

In order to apply the CE algorithm, we need to specify a parameterized random mechanism to generate the random tours. As mentioned, the updating formulas for the parameters follow, as always, from CE minimization.

An easy way to explain how the tours are generated and how the parameters are updated is to relate (8.34) to an *equivalent* minimization problem. Let

$$\widetilde{\mathscr{X}} = \{(x_1, \ldots, x_n) \,:\, x_1 = 1, \quad x_i \in \{1, \ldots, n\}\,, \quad i = 2, \ldots, n\} \tag{8.35}$$

be the set of vectors that correspond to tours that start in 1 and can visit the same city more than once. Note that $|\widetilde{\mathscr{X}}| = n^{n-1}$ and $\mathscr{X} \subset \widetilde{\mathscr{X}}$. When $n = 4$, we could have, for example, $\mathbf{x} = (1, 3, 1, 3) \in \widetilde{\mathscr{X}}$, corresponding to the *path* (not tour) $1 \to 3 \to 1 \to 3 \to 1$. Define the function $\widetilde{S}$ on $\widetilde{\mathscr{X}}$ by $\widetilde{S}(\mathbf{x}) = S(\mathbf{x})$, if $\mathbf{x} \in \mathscr{X}$ and $\widetilde{S}(\mathbf{x}) = \infty$ otherwise. Then, obviously, (8.34) is equivalent to the minimization problem

$$\text{minimize } \widetilde{S}(\mathbf{x}) \text{ over } \mathbf{x} \in \widetilde{\mathscr{X}}\,. \tag{8.36}$$

A simple method to generate a random path $\mathbf{X} = (X_1, \ldots, X_n)$ in $\widetilde{\mathscr{X}}$ is to use a Markov chain on the graph G, starting at node 1 and stopping after n steps. Let $\mathbf{P} = (p_{ij})$ denote the one-step transition matrix of this Markov chain. We assume that the diagonal elements of $\mathbf{P}$ are 0 and that all other elements of $\mathbf{P}$ are strictly positive, but otherwise $\mathbf{P}$ is a general $n \times n$ stochastic matrix.

The pdf $f(\cdot; \mathbf{P})$ of $\mathbf{X}$ is thus parameterized by the matrix $\mathbf{P}$, and its logarithm is given by

$$\ln f(\mathbf{x}; \mathbf{P}) = \sum_{r=1}^{n} \sum_{i,j} I_{\{\mathbf{x} \in \widetilde{\mathscr{X}}_{ij}(r)\}} \ln p_{ij}\,,$$

where $\widetilde{\mathscr{X}}_{ij}(r)$ is the set of all paths in $\widetilde{\mathscr{X}}$ for which the r-th transition is from node i to j. The updating rules for this modified optimization problem follow from (8.18), with $\{S(\mathbf{X}_i) \geqslant \gamma_t\}$ replaced with $\{\widetilde{S}(\mathbf{X}_i) \leqslant \gamma_t\}$, under the condition that the rows of $\mathbf{P}$ sum up to 1. Using Lagrange multipliers $u_1, \ldots, u_n$, we obtain the maximization problem

$$\max_{\mathbf{P}} \min_{u_1, \ldots, u_n} \left\{ \mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \ln f(\mathbf{X}; \mathbf{P}) \right] + \sum_{i=1}^{n} u_i \left(\sum_{j=1}^{n} p_{ij} - 1 \right) \right\}\,. \tag{8.37}$$

Differentiating the expression within braces above with respect to p_{ij} yields, for all $j = 1, \ldots, n$,

$$\frac{\mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X} \in \widetilde{\mathscr{X}}_{ij}(r)\}} \right]}{p_{ij}} + u_i = 0\,. \tag{8.38}$$

Summing over $j = 1, \ldots, n$ gives $\mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X} \in \widetilde{\mathscr{X}}_i(r)\}} \right] = -u_i$, where $\widetilde{\mathscr{X}}_i(r)$ is the set of paths for which the r-th transition starts from node i. It follows that the optimal p_{ij} is given by

$$p_{ij} = \frac{\mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X} \in \widetilde{\mathscr{X}}_{ij}(r)\}} \right]}{\mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X} \in \widetilde{\mathscr{X}}_i(r)\}} \right]}\,. \tag{8.39}$$

The corresponding estimator is

$$\widehat{p}_{ij} = \frac{\displaystyle\sum_{k=1}^{N} I_{\{\widetilde{S}(\mathbf{X}_k)\leqslant\gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X}_k \in \widetilde{\mathscr{X}_{ij}}(r)\}}}{\displaystyle\sum_{k=1}^{N} I_{\{\widetilde{S}(\mathbf{X}_k)\leqslant\gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X}_k \in \widetilde{\mathscr{X}_{i}}(r)\}}} . \tag{8.40}$$

This has a very simple interpretation. To update p_{ij}, we simply take the fraction of times in which the transition from i to j occurs, taking into account only those paths that have a total length less than or equal to γ.

This is how one could, *in principle*, carry out the sample generation and parameter updating for problem (8.36): generate paths via a Markov process with transition matrix $\mathbf{P}$ and use the updating formula (8.40). However, *in practice*, we would never generate the tours this way, since most paths would visit cities (other than 1) more than once, and therefore their $\widetilde{S}$ values would be ∞ — that is, most of the paths would not constitute tours. In order to avoid the generation of irrelevant paths, we proceed as follows:

Algorithm 8.6.1: Trajectory Generation Using Node Transitions

```
Define P^(1) = P and X_1 = 1.
for t = 1 to n − 1 do
    Obtain P^(t+1) from P^(t) by first setting the X_t-th column of P^(t) to 0 and
      then normalizing the rows to sum up to 1.
    Generate X_{t+1} from the distribution formed by the X_t-th row of P^(t).
```

A fast implementation of the algorithm above, due to Slava Vaisman, is given by the following procedure, which has complexity $\mathcal{O}(n^2)$. Here i is the currently visited node, and $(b_1, \ldots, b_n)$ is used to keep track of which states have been visited: $b_i = 1$ if node i has already been visited and 0 otherwise.

Algorithm 8.6.2: Fast Generation of Trajectories.

```
Let b_1 ← 1 and b_j ← 0 for all j ≠ 1.
X_1 ← 1, i ← 1
for t = 1 to n do
    Generate U ∼ U(0, 1) and let R ← U Σ_{j=1}^{n} (1 − b_j) p_ij.
    Let sum ← 0 and j ← 0.
    while sum < R do
        j ← j + 1
        if b_j = 0 then sum ← sum + p_ij
    Set X_t ← j, b_j ← 1 and i ← j.
```

It is important to realize that the updating formula for p_{ij} remains the same. By using Algorithm 8.6.1, we are merely *speeding up* our naive trajectory generation by only generating *tours*. As a consequence, each trajectory will visit each city once, and transitions from i to j can at most occur once. It follows that

$$\widetilde{\mathscr{X}_{ij}}(r) = \widetilde{\mathscr{X}_{i}}(r) = \emptyset, \quad \text{for } r \geqslant 2 ,$$

so that the updating formula for p_{ij} can be written as

$$\widehat{p}_{ij} = \frac{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k)\leqslant\gamma\}}\, I_{\{\mathbf{X}_k\in\mathscr{X}_{ij}\}}}{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k)\leqslant\gamma\}}}\,, \tag{8.41}$$

where $\mathscr{X}_{ij}$ is the set of tours in which the transition from i to j is made. This has the same "natural" interpretation discussed for (8.40).

For the initial matrix $\widehat{\mathbf{P}}_0$, one could simply take all off-diagonal elements equal to $1/(n-1)$, provided that all cities are connected.

Note that ϱ and α should be chosen as in Remark 8.3.3, and the sample size for TSP should be $N = c\,n^2$, with $c > 1$, say $c = 5$.

■ **EXAMPLE 8.11 TSP on Hammersley Points**

To shed further light on the CE method applied to the TSP, consider a shortest (in Euclidean distance sense) tour through a set of *Hammersley points*. These form an example of *low-discrepancy* sequences that cover a d-dimensional unit cube in a pseudo-random but orderly way. To find the 2^5 two-dimensional Hammersley points of order 5, construct first the x-coordinates by taking all binary fractions $x = 0.x_1x_2\ldots x_5$. Then let the corresponding y coordinate be obtained from x by reversing the binary digits. For example, if $x = 0.11000$ (binary), which is $x = 1/2 + 1/4 = 3/4$ (decimal), then $y = 0.00011$ (binary), which is $y = 3/32$ (decimal). The Hammersley points, in order of increasing y, are thus

$$\{(0,0),(16,1),(8,2),(24,3),(4,4),(20,5),(12,6),(28,7),(2,8),(18,9),(10,10),$$
$$(26,11),(6,12),(22,13),(14,14),(30,15),(1,16),(17,17),(9,18),(25,19),(5,20),$$
$$(21,21),(13,22),(29,23),(3,24),(19,25),(11,26),(27,27),(7,28),(23,29),$$
$$(15,30),(31,31)\}/32\,.$$

Table 8.10 and Figure 8.8 show the behavior of the CE algorithm applied to the Hammersley TSP. In particular, Table 8.10 depicts the progression of $\widehat{\gamma}_t$ and S_t^b, which denote the largest of the elite values in iteration t and the best value encountered so far, respectively. Similarly, Figure 8.8 shows the evolution of the transition matrices $\mathbf{P}_t$. Here the initial elements $p_{0,ij}$, $i \neq j$ are all set to $1/(n-1) = 1/31$; the diagonal elements are 0. We used a sample size of $N = 5\,n^2 = 5120$, rarity parameter $\varrho = 0.03$, and smoothing parameter $\alpha = 0.7$. The algorithm was stopped when no improvement in $\widehat{\gamma}_t$ during three consecutive iterations was observed.