Discrete Stochastic Approximation with Application to Resource Allocation

Stacy D. Hill

An optimization problem involves finding the best value of an objective function or figure of merit—the value that optimizes the function. If the set of options is finite in number, then the problem is discrete. If the value of the objective function is uncertain because of measurement noise or some other source of random variation, the problem is stochastic. Mathematically, the discrete optimization problem is a nonlinear optimization problem involving integer variables, and its solution will require some iterative procedure. This article discusses one such procedure for solving difficult optimization problems. The procedure is a discrete-variables version of the Simultaneous Perturbation Stochastic Approximation algorithm, developed at APL, for solving optimization problems involving continuous variables. The discrete-variables algorithm shares some of the computational efficiency of its continuous counterpart.

INTRODUCTION

Discrete optimization problems occur in a wide variety of practical applications. One important class of such problems is the resource allocation problem: There is a finite quantity of some resource that can be distributed in discrete amounts to *users* or to perform a set of tasks; the problem is to distribute the resource so as to optimize some figure of merit or objective function. This article discusses a discrete optimization algorithm for solving such problems.

The algorithm,[1,2] developed in collaboration with László Gerencsér (Computer and Automation Res. Inst., Hungarian Academy of Sciences, Budapest) and Zsuzsanna Vágó (Pázmány Péter Catholic University, Budapest), relies on the method of stochastic approximation (SA).[3] It is a discrete-variables version of an SA method, also developed at APL, called the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm,[4,5] which is used for solving optimization problems involving continuous variables. The goal in developing the discrete version of the SPSA, which we will sometimes call "discrete SPSA," was to design an algorithm that, like its continuous counterpart, is computationally efficient and solves problems in which the objective function is analytically unavailable or difficult to compute. Before presenting the discrete algorithm, several example problems are given to illustrate the variety of discrete resource allocation problems and the need for discrete SPSA. (For other examples, see Ref. 6.)

The first example is the weapons assignment problem.[7] There are multiple weapons systems that differ in number, yield, and accuracy, and there are multiple targets that differ in "hardness" and type. The problem is to assign weapons to targets in some optimal fashion. The resources are weapons assets, and the tasks are the targets to be attacked. The objective function might reflect the cost of deploying assets, target hardness, and the strategic value of each target; it might also reflect the goal to attack a certain minimum number of targets or the requirements to achieve a certain level of damage and minimize undesired collateral damage.

Another example is the facilities location problem[8]: Facilities (e.g., manufacturing plants, military supply bases, schools, warehouses) can be built at a fixed number of locations. There is a cost if the facility is underutilized or if it cannot keep up with the demand for its services. The problem is to determine the best location and size of each facility.

The last example is the problem of scheduling the transmission of messages in a radio network.[9] A message is transmitted over the nodes in a network as a set of "frames," where the total number of frames a message requires depends on the length of the message. There is a fixed number of time slots—buffer space—that can be allocated to message frames. The problem is to allocate buffer space to the nodes to minimize average transmission delays or some other quantity such as the number of messages that are "blocked" or cannot be transmitted.

In each of these examples, the resource can only be distributed in discrete amounts, i.e., the amount of a resource that can be allocated is an integer value. The objective function is some scalar-valued function; depending on its interpretation, the optimal value—and consequently the optimal allocation—corresponds to its minimum or maximum value. For example, if the objective function measures a *loss* such as the cost of an allocation, then an optimal allocation minimizes the loss. If, on the other hand, the objective function measures some gain such as profit or reward, then an optimal allocation is one for which the objective function is at a maximum. In what follows, we will assume that the objective function is a loss function. This assumption imposes no loss in generality, since a maximization problem is easily transformed into one of minimization. More precisely, the problem of finding the minimum of a loss function, L, say, is equivalent to the problem of finding the maximum value of $-L$.

One feature of discrete optimization problems that makes them potentially difficult to solve is the size or cardinality of their search spaces, which can be large in problems involving a relatively small number of users and resources. For example, the number of ways of allocating, say, 20 units of a resource to 30 users exceeds 10^{13}. More generally, the size of the search space for a constrained resource allocation problem consisting of N users and K identical resources (i.e., the resources are indistinguishable) exceeds $(K + N - 1)!/(N - 1)!K!$.[10] Thus, the search space is typically too large to make an exhaustive search a feasible approach.

Adding to the difficulty of dealing with large search spaces is the problem of operating in a "noisy" or stochastic environment. An algorithm for finding the optimal value requires the ability to evaluate the loss function at estimated or candidate solutions. The computed values of the loss will be noisy if the loss function depends on quantities having uncertain values or is corrupted by "measurement" noise. In the weapons assignment problem, for example, some uncertainty may exist in the location or characteristics of the targets or in estimates of damage, and hence the loss may depend on damage assessments obtained by sensor devices that may contain measurement noise. In the facilities location problem, the actual use at a location may vary unpredictably, as will the gain and loss in locating there. In the problem of transmitting messages in a radio network, the loss will be random if users can request network resources at random instants of time or hold them for random lengths of time. Any algorithm for finding the minimum must be applicable to noisy loss functions.

Noisy loss functions and large search spaces present two difficult challenges to solving discrete optimization problems and are the main motivation for the development of a discrete version of SPSA.

PROBLEM FORMULATION

The resource allocation problem is easy to state. Consider the case involving a single type of resource. Suppose K units of the resource are to be distributed to p users or tasks. An allocation rule assigns a fixed number of the units to each user and therefore determines a vector of dimension p—an allocation vector—whose components are the quantities of the resource allocated to users. If θ denotes the allocation vector, then $\theta = (\theta_1, \ldots, \theta_p)$, where θ_j is the amount allocated to the jth user. Since the amounts allocated are discrete, each θ_j is a non-negative integer, and since the total quantity allocated is K, it follows that $\sum_{j=1}^{p} \theta_j = K$. The total loss associated with the allocation θ is $L(\theta)$ and depends on the loss in allocating resources to each of the p users. If $L_j(\theta)$ is the loss associated with the jth user, the total loss is $L(\theta) = \sum_{j=1}^{p} L_j(\theta)$. The problem, then, is to find the allocation that minimizes the total loss, i.e.,

$$\begin{aligned} &\text{minimize } \sum_{j=1}^{p} L_j(\theta) \\ &\text{subject to } \sum_{j=1}^{p} \theta_j = K\,, \end{aligned} \tag{1}$$

where θ_j is a non-negative integer, $j = 1, 2, \ldots, p$. In general terms, this is a nonlinear integer problem—an optimization problem with integer variables and a real-valued objective function.

An allocation vector θ is a *feasible solution* if it satisfies the constraints, i.e., the allocations are non-negative integers and their sum equals K. A feasible solution θ^* is a solution if $L(\theta^*) \leq L(\theta)$ for any other feasible solution θ.

The optimization problem as currently formulated is intractable; that is, any algorithm that solves it must enumerate a nontrivial part of the set of feasible solutions and is essentially equivalent to an exhaustive search. In the language of computational complexity theory, the problem is *NP-complete*. (See Ibaraki and Katoh,[6] pp. 35–37 and 212–214, for a discussion of computational complexity and how it relates to the resource allocation problem.) For this reason, we consider a class of objective functions that lead to tractable problems. In particular, we consider objective functions that are separable and integer convex. The notion of integer convexity will be defined later. The loss function L is separable if

$$L(\theta) = \sum_{j=1}^{p} L_j(\theta_j)\,. \tag{2}$$

This form is a special case of a loss function in which the jth user loss depends only on the allocation to the jth user.

Algorithms for solving an optimization problem are iterative procedures that generate a sequence of estimates that converges to an optimum. These procedures are typically recursive, i.e., the "next" estimate depends on the previous ones. In the deterministic setting—problems in which the loss function can be evaluated at each θ—there are a number of procedures for finding the minimum.[6] For many practical problems, however, uncertainty or noise may exist that makes the loss function difficult or impossible to evaluate.

Algorithms for discrete optimization problems involving noisy loss functions are limited.[10–12] In such problems, the loss values must be replaced by estimates, which may contain measurement noise. (For example, in the facilities location problem, loss depends on the difference between the planned capacity and that which is required to meet user demand, and will therefore be unknown if the actual demand is unpredictable.) The discrete SPSA algorithm, like the SPSA algorithm, is a recursive algorithm in which the next estimate depends in a very simple way on the estimates of the loss function.

Uncertainties that make the loss difficult to evaluate can be viewed as random variables that influence the actual loss, and the (total) loss $L(\theta)$ can be viewed as the average or expected value of the actual loss. More specifically, assume that the uncertain quantities are random variables denoted ω, and denote the actual loss by $\ell(\theta, \omega)$; then

$$L(\theta) = \text{the expected value of } \ell(\theta, \omega)\,.$$

Similarly, if the actual loss for the jth user is $\ell_j(\theta, \omega)$, then

$$L_j(\theta) = \text{the expected value of } \ell_j(\theta, \omega)$$

and

$$\ell(\theta, \omega) = \sum_{j=1}^{p} \ell_j(\theta, \omega)\,.$$

Another way of viewing the actual and expected losses is to think of the actual loss as a measurement of the expected loss that is corrupted by additive noise. In other words, if the measurement noise is $\varepsilon_j(\theta, \omega)$ and has zero mean, then

$$\ell_j(\theta, \omega) = L_j(\theta) + \varepsilon_j(\theta, \omega)\,. \tag{3}$$

Thus $\ell_j(\theta, \omega)$ is a noisy measurement of $L_j(\theta)$. Likewise, the total actual loss, $\ell(\theta, \omega)$, is a noisy measurement of $L(\theta)$, the total expected loss.

Since the loss function is not directly available, the optimization algorithm must rely on noisy estimates of the loss for finding the minimum.

THE OPTIMIZATION ALGORITHM

The discrete SPSA method is an analogue of the SPSA algorithm for continuous-variables problems. Let us briefly review the SPSA algorithm to see how it is modified to obtain the discrete version.

Continuous-Variables SPSA

In the continuous setting, θ is a continuous vector parameter, i.e., its components are real numbers. The optimization problem is

$$\begin{aligned}&\text{minimize } L(\theta)\,,\\ &\text{where } \theta_j \text{ is a real number, } j = 1, 2, \ldots, p\,.\end{aligned} \tag{4}$$

The loss function L is assumed to be a differentiable real-valued function. Thus, if a point θ^* minimizes L, then

$$\frac{\partial L(\theta^*)}{\partial \theta} = 0\,. \tag{5}$$

Under additional assumptions, the root of this equation minimizes L.

As in the discrete optimization problem, the loss function is assumed to be unavailable. However, one can obtain noisy measurements of the loss,

$$y(\theta) = L(\theta) + \varepsilon(\theta, \omega) ,$$

where ε is measurement noise and, as before, ω denotes uncertainty. Since $L(\theta)$ is unavailable, its gradient, $\partial L/\partial\theta$, is also unavailable and must be estimated.

The SPSA algorithm for solving Eq. 5 uses a computationally efficient estimate of the gradient, which is computed in terms of the $y(\theta)$ values, the noisy observations of the loss. The algorithm is

$$\hat{\theta}_{k=1} = \hat{\theta}_k - a_k \hat{g}(\hat{\theta}_k) . \tag{6}$$

For $k = 1, 2, 3, \ldots$, the *gain sequence* a_k is an appropriately chosen sequence of positive real numbers, and $\hat{g}(\hat{\theta}_k)$ is an estimate of the gradient $g(\theta) = \partial L(\theta)/\theta$ of the loss function evaluated at $\hat{\theta}_k$ defined as follows:

Step 1. Generate a vector $\Delta_k = (\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kp})$, the components of which are Bernoulli random variables taking the values ± 1 with probability 1/2.

Step 2. Take a positive real number c_k, the *step size*, and consider the two perturbations about $\hat{\theta}_k$:

$$\hat{\theta}_k^{(+)} = \hat{\theta}_k + c_k \Delta_k$$

and

$$\hat{\theta}_k^{(-)} = \hat{\theta}_k - c_k \Delta_k .$$

(Spall,[13] pp. 189–190, contains guidelines for choosing values for c_k and a_k.)

Step 3. Evaluate y at the perturbed values $\hat{\theta}_k^{(+)}$, $\hat{\theta}_k^{(-)}$ to obtain $y_k^{(+)} = y(\hat{\theta}_k^{(+)})$ and $y_k^{(-)} = y(\hat{\theta}_k^{(-)})$. These are measurements of $L(\hat{\theta}_k^{(+)})$ and $L(\hat{\theta}_k^{(-)})$, respectively.

Step 4. Form the estimate $\hat{g}(\hat{\theta}_k)$ by taking

$$\hat{g}(\hat{\theta}_k) = \frac{(y_k^{(+)} - y_k^{(-)})}{2c_k} \begin{bmatrix} \dfrac{1}{\Delta_{k1}} \\ \dfrac{1}{\Delta_{k2}} \\ \vdots \\ \dfrac{1}{\Delta_{kp}} \end{bmatrix} . \tag{7}$$

Under suitable conditions on the loss function, the estimates $\hat{\theta}_k$ converge to a solution of Eq. 5.

The gradient estimate requires, at each iteration, only *two* measurements of the loss, namely, $y_k^{(+)}$ and $y_k^{(-)}$. The standard method of estimating the gradient from observations, the method of finite differences, requires at least $p + 1$ measurements of the loss. The SPSA is computationally efficient compared with an SA algorithm that uses finite difference, especially if loss measurements are time-consuming or costly to obtain. It is this type of efficiency that is sought for the discrete algorithm.

Discrete Parameter SPSA

A Special Case

The discrete algorithm is similar to the continuous algorithm; however, in the discrete setting there is no derivative. Under suitable conditions, differences between the loss function at different points behave like derivatives in the continuous-variables setting. One condition that guarantees this behavior is a convexity condition for functions of a discrete argument[6,14,15] and leads to a discrete-variables analogue of the SPSA algorithm.

Before exploring the notion of discrete convexity, it may be helpful to review convexity in the continuous setting. Geometrically, a function is convex if, at each point on its graph, there is a line which passes through that point which lies on or below the graph. Any such line is called a *line of support*. For example, in Fig. 1, the solid curve (blue) is the graph of a convex function and the dashed line (red) is a line of support.

The integer convexity condition is not too difficult to describe in the one-dimensional case, where the loss function is a function of a single integer variable. In this instance, the loss function is said to be *integer convex* or simply *convex* if, for each integer θ,

$$L(\theta) \le \frac{L(\theta+1) + L(\theta-1)}{2} . \tag{8}$$

This condition is similar to mid-point convexity for functions of a real argument, where mid-point convexity and convexity are equivalent. An equivalent form of the previous inequality is

$$L(\theta) - L(\theta-1) \le L(\theta+1) - L(\theta) . \tag{9}$$

It is this last inequality that motivates the use of differences as a replacement for derivatives.

To see the similarity between differences and gradients, we need the following fact about integer convex functions. Let $\Delta L(\theta) = L(\theta) - L(\theta - 1)$. A point θ^* minimizes an integer convex function L if

$$\Delta L(\theta^*) \le 0 \le \Delta L(\theta^* + 1) . \tag{10}$$

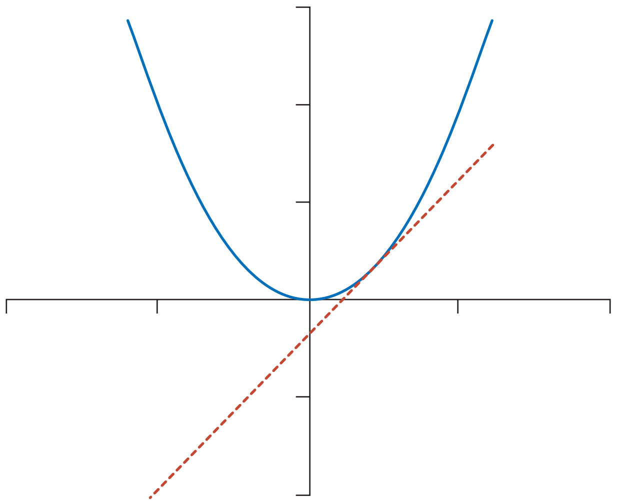

Figure 1. Continuous convex function.

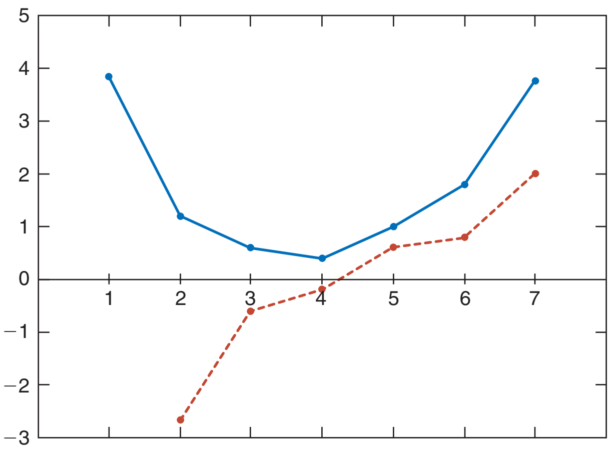

Figure 2. Integer convex function.

Figure 2 illustrates this property and also the connection between $\Delta L(\theta)$ and the loss function. The blue dots, which are connected by the solid line, plot the values of the loss function, and the red dots, which are connected by the dashed line, plot the values of $\Delta L(\theta)$. The minimum of the function occurs at $\theta^* = 4$.

If we think of differences $\Delta L(\theta)$, with θ being an integer, as the discrete analogue of a gradient, then this last inequality implies that to find the minimum of L, we need only look for the point at which the gradients $\Delta L(\theta)$ and $\Delta L(\theta + 1)$ are "close" to zero, or, equivalently, the point at which their average $(\Delta L(\theta + 1) + \Delta L(\theta))/2$ is close to zero.

Observe that

$$\frac{\Delta L(\theta+1)+\Delta L(\theta)}{2} = \frac{L(\theta+1)-L(\theta-1)}{2} . \qquad (11)$$

So the problem of minimizing L reduces to the problem of finding the point at which the discrete "gradient,"

$$g(\theta) = \frac{L(\theta+1)-L(\theta-1)}{2} , \qquad (12)$$

is zero or close to zero.

The quantity $g(\theta)$ behaves very much like a gradient. To see this, we need to extend the loss function to a function of a continuous variable. Consider the function $\bar{L}(\theta)$ obtained by linearly interpolating L between $\theta - 1$ and θ, $\theta = 1, 2, 3, \ldots$. The extension $\bar{L}(\theta)$ is defined for each real number θ. This function is a continuous convex function, but is not everywhere differentiable. Furthermore, if the step size c is small enough, where $c > 0$, then

$$\begin{aligned}\frac{\bar{L}(\theta+c\cdot 1)-\bar{L}(\theta-c\cdot 1)}{2c} &= \frac{1}{2}(L(\theta+1)-L(\theta-1)) \\ &= \frac{1}{2}(L(\theta+1)-L(\theta)) \qquad (13) \\ &\quad + \frac{1}{2}(L(\theta)-L(\theta-1))\end{aligned}$$

if θ is an integer and

$$\frac{\bar{L}(\theta+c\cdot 1)-\bar{L}(\theta-c\cdot 1)}{2c} = L([\theta]+1)-L([\theta]) \qquad (14)$$

if θ is not an integer, where $[\theta]$ denotes the integer part of θ. In either instance, denote the difference by $g(\theta)$, so that

$$g(\theta) = \begin{cases} \frac{1}{2}(L(\theta+1)-L(\theta-1)), & \theta = \text{an integer} \\ L([\theta]+1)-L([\theta]), & \text{otherwise}. \end{cases} \qquad (15)$$

Then $g(\theta)$ behaves, in some sense, like a gradient for extended function $\bar{L}(\theta)$. In fact, g is a *subgradient* and serves as a gradient in the optimization of functions that are not necessarily differentiable.

An estimate of the subgradient is

$$\hat{g}(\theta) = \frac{\bar{y}(\theta+c\Delta)-\bar{y}(\theta-c\Delta)}{2c\Delta} , \qquad (16)$$

where $\bar{y}(\theta)$ is the (piecewise linear) extension of $y(\theta)$, and Δ is a Bernoulli random variable taking the values ± 1 with equal probability. This estimate is a simultaneous perterbation estimate of the subgradient.[2]

The foregoing suggests an SA algorithm that is analogous to the continuous version (Eq. 6):

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}([\hat{\theta}_k]), \quad \hat{\theta}_1 = \text{an integer} . \qquad (17)$$

If the algorithm is stopped at iteration n, the estimate of the optimizing value is $[\hat{\theta}_n]$. Note that this algorithm is similar to the discrete SA algorithm in Ref. 16.

The General Algorithm

Assume that the loss function is separable and convex, i.e.,

$$L(\theta) = \sum_{j=1}^{p} L_j(\theta_j) , \tag{18}$$

where each L_j is an integer convex function. Similar to Inequality 10, the point θ^* minimizes this loss function if

$$\Delta L_j(\theta_j^*) \le 0 \le \Delta L_j(\theta_j^* + 1),\ j = 1,\ \ldots, p . \tag{19}$$

Unlike the one-dimensional case, this condition is sufficient but not necessary for a minimum. (Cassandras et al.[10] give sufficient conditions for a minimum.)

Let $\Delta L(\theta) = (\Delta L_1(\theta_1), ..., \Delta L_p(\theta_p))$. If we view $\Delta L(\theta)$ and $\Delta L(\theta + 1)$ as gradients of L, then the minimum occurs at a point where their average $(\Delta L(\theta + 1) + \Delta L(\theta))/2$ is close to zero. Let

$$g(\theta) = \frac{\Delta L(\theta+1) + \Delta L(\theta)}{2} .$$

Again, the problem of minimizing the loss reduces to finding a point at which g is zero or close to zero.

Since L is separable, g satisfies the following identity:

$$g(\theta) = \left(\frac{L_1(\theta_1+1) - L_1(\theta_1-1)}{2}, \ldots, \frac{L_p(\theta_p+1) - L_p(\theta_p-1)}{2} \right) . \tag{20}$$

This form of g and the discussion in the last section suggest the following simple "gradient" estimate similar to the estimate in SPSA. Let $\Delta_k = (\Delta_{k1}, \Delta_{k2}, ..., \Delta_{kp})$, where the components of Δ_k are independent Bernoulli random variables taking the values ± 1. Then

$$\hat{g}(\hat{\theta}_k) = \frac{(y(\hat{\theta}_k + \Delta_k) - y(\hat{\theta}_k - \Delta_k))}{2} \begin{bmatrix} \dfrac{1}{\Delta_{k1}} \\ \dfrac{1}{\Delta_{k2}} \\ \vdots \\ \dfrac{1}{\Delta_{kp}} \end{bmatrix} . \tag{21}$$

This estimate satisfies an important property: it is an unbiased estimate of the subgradient.

The subgradient estimate in Eq. 21 leads to an algorithm similar to the one in Eq. 17:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}([\hat{\theta}_k]) . \tag{22}$$

The initial value is $\hat{\theta}_1 = (\hat{\theta}_{11}, \hat{\theta}_{12}, \ldots, \hat{\theta}_{1p})$, where each $\hat{\theta}_{1j}$ is an integer, $j = 1, \ldots, p$, and $[\theta]$ is the vector obtained by taking the integer part of each component of θ. To solve the resource allocation problem, this algorithm requires a slight modification that ensures that each iterate is a feasible solution. An easy way to guarantee this is by a projection—if an iterate lands outside the feasible solution set, *project* it onto the feasible solution set before generating the next iterate.

The steps to implement the discrete algorithm are similar to those for the continuous one:

Step 1. Generate a vector $\Delta_k = (\Delta_{k1}, \Delta_{k2}, ..., \Delta_{kp})$, the components of which are Bernoulli random variables taking the values ± 1 with probability 1/2.

Step 2. Form the value $[\hat{\theta}_k]$, the vector with integer components.

Step 3. Consider the two perturbations about $[\hat{\theta}_k]$:

$$[\hat{\theta}_k]^{(+)} = [\hat{\theta}_k] + \Delta_k$$

and

$$[\hat{\theta}_k]^{(-)} = [\hat{\theta}_k] - \Delta_k .$$

Step 4. Evaluate y at the perturbed values $[\hat{\theta}_k]^{(\pm)}$ to obtain $y_k^{(+)} = y(\hat{\theta}_k{}^{(+)})$ and $y_k^{(-)} = y(\hat{\theta}_k{}^{(-)})$ (measurements of $L(\hat{\theta}_k{}^{(+)})$ and $L(\hat{\theta}_k{}^{(-)})$, respectively) and form the estimate of $\hat{g}(\hat{\theta}_k)$.

Step 5. Update the algorithm according to Eq. 22.

DISCUSSION

The performance of the algorithm is an important practical issue. Its numerical performance has been studied previously.[1,17] It has also been compared (see Ref. 18) with the method of simulated annealing,[11] an algorithm that applies to functions of a discrete argument. Numerical studies provide insight into specific applications and performance in special problems when compared with existing algorithms. However, numerical studies do not prove convergence, which is a requirement of any algorithm. Convergence is a theoretical property and must be established by a proof of convergence, as has been done for the continuous algorithm.[4,19]

With respect to convergence, there are two important questions: Does the algorithm converge? How fast does it converge? The discrete SPSA algorithm (Eq. 22) converges under some restrictions on the loss function.[2] The latter question about the rate of convergence remains open and is currently under investigation.

REFERENCES

[1]Gerencsér, L., Hill, S. D., and Vágó, Z., "Optimization over Discrete Sets via SPSA," in *Proc. 38th IEEE Conf. on Decision and Control,* Phoenix, AZ, pp. 1791–1795 (1999).

[2]Hill, S. D., Gerencsér, L., and Vágó, Z., "Stochastic Approximation on Discrete Sets Using Simultaneous Difference Approximations," in *Proc. 2004 Am. Control Conf.*, Boston, MA, pp. 2795–2798 (2004).

[3]Kushner, H. J., and Clark, D. S., *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, Springer-Verlag (1978).

[4]Spall, J. C., "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Trans. Auto. Contr.* **37**, 332–341 (1992).

[5]Spall, J. C., "An Overview of the Simultaneous Perturbation Method for Efficient Optimization," *Johns Hopkins APL Tech. Dig.* **19**(4), 482–492 (1998).

[6]Ibaraki, T., and Katoh, N., *Resource Allocation Problems: Algorithmic Approaches*, MIT Press (1988).

[7]Cullenbine, C. A., Gallagher, M. A., and Moore, J. T., "Assigning Nuclear Weapons with Reactive Tabu Search," *Mil. Operations Res.* **8**, 57–69 (2003).

[8]Ermoliev, Y., "Facility Location Problem," in *Numerical Techniques for Stochastic Optimization*, Y. Ermoliev and R. J-B. Wets (eds.), Springer, New York, pp. 413–434 (1988).

[9]Wieselthier, J. E., Barnhart, C. M., and Ephremides, A., "Optimal Admission Control in Circuit-Switched Multihop Networks," in *Proc. 31st IEEE Conf. on Decision and Control*, Tucson, AZ, pp. 1011–1013 (1992).

[10]Cassandras, C. G., Dai, L., and Panayiotou, C. G., "Ordinal Optimization for a Class of Deterministic and Stochastic Discrete Resource Allocation Problems," *IEEE Trans. Auto. Contr.* **43**(7), 881–900 (1998).

[11]Gelfand, B., and Mitter, S. K., "Simulated Annealing with Noisy or Imprecise Energy Measurements," *J. Optimiz. Theory App.* **62**, 49–62 (1989).

[12]Kleywegt, A., Homem-de-Mello, T., and Shapiro, A., "The Sample Average Approximation Method for Stochastic Discrete Optimization," *SIAM J. Optimiz.* **12**(2), 479–502 (2001).

[13]Spall, J. C., *Introduction to Stochastic Search and Optimization*, Wiley, NJ (2003).

[14]Favati, P., and Tardella, F., "Convexity in Nonlinear Integer Programming," *Ric. Operat.* **53**, 3–34 (1990).

[15]Miller, B. L., "On Minimizing Nonseparable Functions Defined on the Integers with an Inventory Application," *SIAM J. App. Math.* **21**, 166–185 (1971).

[16]Dupac, V., and Herkenrath, U., "Stochastic Approximation on a Discrete Set and the Multi-Armed Bandit Problem," *Comm. Stat. Sequential Anal.* **1**, 1–25 (1982).

[17]Whitney, J. E. II, Hill, S. D., and Solomon, L. I., "Constrained Optimization over Discrete Sets via SPSA with Application to Non-Separable Resource Allocation," in *Proc. 2001 Winter Simulation Conf.*, Arlington, VA, pp. 313–317 (2001).

[18]Whitney, J. E. II, Hill, S. D., Wairia, D., and Bahari, F., "Comparison of the SPSA and Simulated Annealing Algorithms for the Constrained Optimization of Discrete Non-Separable Functions," in *Proc. 2003 Am. Control Conf.*, Denver, CO, pp. 3260–3262 (2003).

[19]Gerencsér, L., "Rate of Convergence of Moments for a Simultaneous Perturbation Stochastic Approximation Method for Function Minimization," *IEEE Trans. Auto. Contr.* **44**, 894–906 (1999).

ACKNOWLEDGMENT: This work was partially supported by the APL IR&D program.

THE AUTHOR

Stacy D. Hill joined APL in 1983 and is a member of the Strategic Systems Department. He received a B.S. in psychology and an M.S. in mathematics in 1975 and 1977, respectively, both from Howard University, and a D.Sc. in engineering (systems science and mathematics) in 1982 from Washington University in St. Louis. Dr. Hill has led a variety of systems analysis and modeling projects, including developing techniques for analyzing weapons systems accuracy performance. He has published articles on diverse topics, including simulation, optimization, and parameter estimation. His article, "Optimization of Discrete Event Dynamic Systems via Simultaneous Perturbation Stochastic Approximation," was published in a special issue of *IIE Transactions* and received a Best Paper award. His current research interest is stochastic discrete optimization. Dr. Hill can be reached via e-mail at stacy.hill@jhuapl.edu.

Stacy Hill