Solving Discrete Resource Allocation Problems using the Simultaneous Perturbation Stochastic Approximation (SPSA) Algorithm

Otis Brooks
Sr. Professional Staff, Aviation Systems Engineering Group
The Johns Hopkins University Applied Physics Laboratory, Laurel, MD 20723, USA
otis.brooks@jhuapl.edu

Keywords: Resource Allocation, Discrete Stochastic Optimization, Simultaneous Perturbation Stochastic Approximation

Abstract

We investigate optimization techniques for solving a class of discrete resource allocation problems, including several discrete forms of Simultaneous Perturbation Stochastic Optimization (SPSA). We explore the rate-of-convergence for discrete SPSA in a stochastic setting. Finally, we consider some of the difficulties that can arise when discrete resource allocation problems include a stochastic component.

1. INTRODUCTION

The aim of this paper is to add to the literature in the area of discrete stochastic optimization by comparing and analyzing the performance of two stochastic optimization algorithms when applied to discrete resource allocation problems. The notion of feasible allocations at every iteration becomes important in this setting and so a projection algorithm is also introduced to ensure feasibility for discrete SPSA.

The rest of the paper is organized as follows: In Section 2, we introduce the facility location and size problem as the example resource allocation problem that we examine. Section 3 introduces the optimization algorithms that we will analyze, Ordinal Optimization and discrete SPSA. Section 4 describes the numerical study undertaken in this project. Section 5 discusses the results of the numerical study. Finally, Section 6 presents the conclusions of the study.

2. THE GENERAL RESOURCE ALLOCATION PROBLEM

2.1. Background

The general resource allocation problem involves the distribution of a finite quantity of resources to users in order to accomplish some task. We wish to distribute the resources in such a way that some measure of performance is optimized. The problems of scheduling the transmission of messages in a radio network, weapon assignment and facility location typify this class of problems.

From [1], we get the data for this study. The resources turn out to be a total of *T* students requiring an education in a metropolitan area where the students are distributed over *K* districts. The *N* user classes turn out to be *N* schools. It goes without saying that resources are allocated in discrete amounts.

We will use this data to fashion an interesting resource allocation problem involving stochasticity.

2.2. Stochasticity

Suppose the students chose schools according to some distribution unknown to school system planners. In this study, we used data from [1] which involves a high school location problem in Turin, Italy (the city is divided into 23 school districts with one school for each district) to deduce a probability matrix, essentially a state transition matrix, which describes the stochastic nature of this problem.

Suppose x_j is the planned size of the school at school location *j*, then the actual number of students, τ_j, attracted to school j may not be equal to x_j. We can cast an interesting stochastic optimization problem around the determination of the optimal size of the schools $j = 1 \ldots N$.

From [1], consider the following cost function which quantifies a cost associated with the building being too large $\left(\alpha_j^+\right)$ for incremental size x_j greater than the (stochastic) demand τ_j and a cost associated with the building being too small $\left(\alpha_j^-\right)$ for incremental size x_j less than the (stochastic) demand τ_j:

$$f_j\left(x_j, \tau_j\right) = \begin{cases} \alpha_j^+\left(x_j - \tau_j\right), & \text{if } x_j \geq \tau_j, \\ \alpha_j^-\left(\tau_j - x_j\right), & \text{if } x_j < \tau_j, \end{cases}$$

in which x_j represents a guess at an optimal facility size at location j and τ_j represents a Monte Carlo determined instance of demand on the facility at location *j*. So a penalty is assessed based on whether the resulting school capacity is too large or too small.

The direct determination of the distribution of τ_j is practically quite difficult in this case. Instead, random realizations can be generated by simulating individual

choices of students according to the probabilities P_{ij} which are the probabilities that students in district *i* will choose the facility at location *j*, defined as:

$$P_{ij} = \frac{e^{-\lambda c_{ij}}}{\sum_{j=1}^{N} e^{-\lambda c_{ij}}}$$

where λ is a constant and c_{ij} are empirical coefficients that depend on the distance between *i* and *j* (in this case: travel times in minutes).

By applying inverse-transform methodology, random samples of integers behaving according to P_{ij} can be generated from the uniform distribution, *U*, using the following rules:

Suppose a student from district *i* can choose to go to a school at school location *j* with probability, P_{ij}. Then the student has 23 possibilities governed by the P_{ij}'s with $\sum_{j=1}^{N} P_{ij} = 1$ for all *i*'s. Then for each instance of a student from district *i*:

$$\text{School of choice will be} = \begin{cases} \text{School}_1 & \text{if } 0 \le U \le P_{i1} \\ \text{School}_1 & \text{if } P_{i1} < U \le P_{i1} + P_{i2} \\ \vdots & \vdots \\ \text{School}_N & \text{if } P_{i1} + \cdots + P_{iN-1} < U \le 1 \end{cases}$$

The resulting stochastic programming problem is then as follows: determine the sizes x_j of the facilities $j = 1 \ldots N$ that minimizes the expected cost:

$$F(x_1 \ldots x_N) = \sum_{j=1}^{N} E f_j(x_j, \tau_j)$$

subject to constraints:

$$0 \ge x_j \ge T \qquad j = 1 \ldots N, x_j \text{ an integer}$$

$$\sum_{j=1}^{N} s_{ij} = a_i \qquad i = 1 \ldots K$$

$$\sum_{i=1}^{K} a_i = T$$

where s_{ij} is the expected flow of students from district *i* to school location *j* $(i = 1 \ldots K, j = 1 \ldots N)$ per unit time and a_i is the total demand (in terms of students to be taught per unit time) at each district *i*. We can reformulate the above into the stochastic minmax problem with the new objective function:

$$F(X) = \sum_{j=1}^{N} E \max\left[\alpha_j^+(x_j - \tau_j), \alpha_j^-(\tau_j - x_j)\right]$$

We have an *N*-dimensional vector representing an allocation instance of incremental facility size for all districts:

$$X = \left[x_1, x_2, \ldots x_N\right]^T$$

We have an *N*-dimensional vector representing an instance (stochastic) of demand on the facility for all districts:

$$\tau = \left[\tau_1, \tau_2, \ldots \tau_N\right]^T$$

Thus, this optimization problem is discrete because the set of options from which to choose is finite in number. It is stochastic because there is some uncertainty in the approximation of the value of the objective function.

There are no destination constraints (maximum school size) in this problem.

3. OPTIMIZATION ALGORITHMS

Two optimization schemes will be employed to solve the above stochastic integer programming problem, specifically:

- ◊ Ordinal Optimization
- ◊ Discrete SPSA

3.1. Ordinal Optimization

Ordinal Optimization comes to us from [2]. A key feature of this algorithm is that it utilizes ordinal estimates which Cassandras et al. states is particularly robust with respect to estimation noise compared to cardinal estimates. Gerencser et al. asserts in [5] that Ordinal Optimization is a "relaxation-type algorithm in which at any time the allocation is rebalanced between exactly two tasks," or, in our case, users.

Ordinal Optimization requires that the objective function be separable and convex. It can be shown that our stochastic minmax objective function is both separable and convex.

3.2. Discrete Simultaneous Perturbation Stochastic Approximation (SPSA)

SPSA was originally developed to solve continuous parameter optimizations (Spall [7]). It is essentially a Kiefer-Wolfowitz stochastic approximation scheme that relies on an efficient "simultaneous perturbation" approximation of the objective function gradient $g(\theta) \equiv \partial L(\theta) / \partial \theta$.

The SPSA procedure is based on simultaneous random perturbations to estimate the gradient of a loss function by computing differences. At each iteration of the algorithm, we generate a random perturbation vector $\Delta_k = \left[\Delta_{k1}, \ldots, \Delta_{kp}\right]^T$, where the Δ_{ki}'s form an i.i.d sequence of Bernoulli random variables taking the values ± 1. The perturbations are assumed to be independent of the

measurement noise process. For cost functions defined on $\mathbb{R}^p$, the difference estimates at iteration *k* is obtained by evaluating $y_k(\cdot)$ at the two values:

$$y_k^+(\theta,c) = L(\theta + c\Delta_k) + \varepsilon_{2k-1}(\theta + c\Delta_k)$$

$$y_k^-(\theta,c) = L(\theta - c\Delta_k) + \varepsilon_{2k}(\theta - c\Delta_k)$$

with the *i*-th component of the difference estimate being:

$$\hat{g}_{ki}(\theta,c) = \frac{\left(y_k^+(\theta,c) - y_k^-(\theta,c)\right)}{2c\Delta_{ki}}$$

Resulting in the recursion:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k\left(\hat{\theta}_k, c_k\right)$$

From [3] and [4], we are presented with discrete forms of SPSA. These discrete forms of SPSA can be characterized by how the algorithm updates the iterate vector $\hat{\theta}_k$, how it employs the iterate vector in its determination of $\hat{g}_k\left(\hat{\theta}_k\right)$ and what is the nature of c_k and a_k. Essentially, either $\hat{\theta}_k$, $\hat{g}_k\left(\hat{\theta}_k\right)$, $a_k\hat{g}_k\left(\hat{\theta}_k\right)$ or $\hat{\theta}_k - a_k\hat{g}_k\left(\hat{\theta}_k\right)$ are constrained to lie in the set $\mathbb{Z}^p$, the grid of points in $\mathbb{R}^p$ with integer components. We will restrict ourselves to the following instantiations of discrete SPSA:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k\left(\mathrm{Proj}\left(\hat{\theta}_k\right), c_k\right)$$

$$\hat{\theta}_{k+1} = \mathrm{Proj}\left(\mathrm{ProbMove}\left(\hat{\theta}_k - a_k\hat{g}_k\left(\hat{\theta}_k, c_k\right)\right)\right)$$

where $\mathrm{Proj}(\cdot)$ is a mapping of the iterate $\hat{\theta}_k$ to a hypercube of feasible allocations about $\hat{\theta}_k$ (more below) and $\mathrm{ProbMove}(\cdot)$ is a "probabilistic movement" scheme devised to get an iterate $\hat{\theta}_k$ not necessarily an integer to "move" to a point in $\mathbb{Z}^p$ according to the following:

If $\hat{\theta}_k = \left[\hat{\theta}_{k1}, \ldots, \hat{\theta}_{kp}\right] \in \mathbb{R}^p$, then the ith component of $\mathrm{ProbMove}\left(\hat{\theta}_k\right)$ equals $\lfloor \hat{\theta}_{ki} \rfloor$ with probability $\pi_i = \left(\hat{\theta}_{ki} - \lfloor \hat{\theta}_{ki} \rfloor\right) \Big/ \left(\lceil \hat{\theta}_k \rceil - \lfloor \hat{\theta}_{ki} \rfloor\right)$ and $\lceil \hat{\theta}_{ki} \rceil$ with probability $1 - \pi_i$, $1 \le i \le p$.

In the first discrete SPSA scheme, iterates are, in general, not integers. Therefore, we must employ the $\mathrm{Proj}(\cdot)$ mapping when considering the output of the recursion at any index *k* to ensure ourselves of a feasible allocation. In the second scheme, iterates are always integers and the allocation is always feasible.

3.2.1. Fixed Versus Variable Gains

Gerencser et al. suggests the use of fixed gains $(a_k \text{ and } c_k)$ to handle discrete SPSA in [5]. Indeed, much of the literature suggests that constant gain Stochastic Approximation techniques yield reasonable results in discrete optimization problem classes such as our resource allocation problem. We consider this approach but we also consider the customary variable gain process defined in Spall [7], which involve a certain sequence of gains a_k and c_k that approach zero according to the requirements:

$$a_k \text{ and } c_k > 0; a_k \text{ and } c_k \to 0; \sum_{k=0}^{\infty} a_k = \infty \text{ and } \sum_{k=0}^{\infty} a_k^2 \Big/ c_k^2 < \infty$$

These requirements are met via $a_k = a/(k+1+A)^{\alpha}$ and $c_k = c/(k+1)^{\gamma}$; a, A, c, α, and γ are chosen in accordance with Spall [7]. We even consider combinations of the two such as a fixed c_k and a $a_k \to 0$.

3.2.2. Feasible Allocations at Every Iteration

For this study, feasibility must be maintained with each iteration due to the goal to optimize some resource allocation when the cost function is not available in closed form. Hence, no *a priori* loss and change in loss information is available "offline." We intend these algorithms to be "online." So, we use estimates of loss and change in loss information over some observation period and we iterate over each observation period by adjusting the allocation, which requires the allocation to remain feasible with each period. Ordinal Optimization maintains feasibility as an inherent feature of the algorithm through this notion of a "relaxation-type algorithm in which at any time the allocation is rebalanced between exactly two tasks."

3.2.3. $\mathrm{Proj}(\cdot) \to$ Projections Over Multiple Dimensions

Given candidate allocations derived through an iterative process such as SPSA, feasibility is not assured (unlike the Ordinal Optimization algorithm in which allocations are rebalanced between exactly two tasks). Hence, some measures must be taken to maintain feasibility throughout the iterations. We can employ the process of mathematical projection to accomplish this. We are given a $\hat{\theta}^{(k)} = \left[\hat{n}_1^{(k)}, \ldots, \hat{n}_N^{(k)}\right]$ in which there is a non-zero probability that $\sum_{i=1}^{N} \hat{n}_i^{(k)} \neq K$ for some iteration *k*. In such a case, we project $\hat{\theta}^{(k)}$ to the plane represented by $\sum_{i=1}^{N} \hat{n}_i^{(k)} = K$. Linear algebra provides us with the notion of an orthogonal spanning set $S = \left\{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_q\right\}$ of nonzero vectors which

generates a subspace (say v_0) of a vector space $\mathbb{R}^p$ (or, in our case $\mathbb{Z}^p$). If the vector space includes an inner product, then an orthogonal projection P_0 onto the subspace v_0 is defined as:

$$P_0 v_0 = \alpha_1 \mathbf{v}_1 + \cdots + \alpha_q \mathbf{v}_q, \quad \text{where } \alpha_i = \frac{(\mathbf{v}_i, \mathbf{v})}{(\mathbf{v}_i, \mathbf{v}_i)}.$$

For the case in which $\sum_{i=1}^{N} \hat{n}_i^{(k)} = K$, for $K \neq 0$, we do not have a subspace and the above process is not applicable. Therefore, we translate from $\sum_{i=1}^{N} \hat{n}_i^{(k)} = K$ to $\sum_{i=1}^{N} \hat{n}_i^{(k)} = 0$, perform our projection to the subspace $\sum_{i=1}^{N} \hat{n}_i^{(k)} = 0$ and translate back to $\sum_{i=1}^{N} \hat{n}_i^{(k)} = K$. Linearity of the translation allows us to be able to utilize this technique. The challenge here is in generating the orthogonal spanning set *S* for the subset we seek to generate is the subspace of $\mathbb{Z}^{23}$ of all solutions to $\sum_{i=1}^{N} \hat{n}_i = 0$. One trivial spanning set that would generate a subspace of $\mathbb{Z}^{23}$ of all solutions to $\sum_{i=1}^{N} \hat{n}_i = 0$ is the set of 22 vectors of the form:

$$\left.\begin{array}{l} \begin{bmatrix} -1 & 0 & 0 & \cdots & 1 \end{bmatrix}^T \\ \begin{bmatrix} -1 & 0 & 0 & \cdots 1 & 0 \end{bmatrix}^T \\ \begin{bmatrix} -1 & 0 & \cdots 1 & 0 & 0 \end{bmatrix}^T \\ \qquad\quad \vdots \\ \begin{bmatrix} -1 & 1 & \cdots & 0 & 0 \end{bmatrix}^T \end{array}\right\} \text{23 Columns for each vector}$$

Though technically a basis, this spanning set does not suit our needs. However, Matlab provides us with a function that creates an orthogonal basis given another basis as an input. This function was used in this study with great effect.

4. NUMERICAL STUDY

4.1. Purpose & Goals

We will compare Discrete SPSA with Ordinal Optimization noting the accuracy of optimal allocations with respect to optimal solutions identified in [1] which are based on the mean and median of the probability distributions of the 23 school districts. Additionally, we will investigate the asymptotic behavior and distribution of the discrete SPSA iterates and arrive at a rate of convergence figure. In order to do this, we characterize Discrete SPSA with respect to the continuous version of SPSA, for which there is a wealth of knowledge on asymptotic behavior and convergence.

4.2. Ordinal Optimization

The Ordinal Optimization algorithm of [2] was implemented in Matlab code and executed repeatedly to check out implementation. One hundred runs were accomplished to develop a statistically relevant dataset.

The observed sample path length $f(k)$ (number of observations of $\Delta\tilde{L}\left(\tilde{n}^{(k)}\right)$ at each step *k*) was set to 4. This resulted in the best overall performance of the Ordinal Optimization algorithm for the cost function in question.

4.3. Continuous SPSA

Matlab code implemented the basic SPSA algorithm of [7] with relative ease. Again 100 runs were accomplished.

4.3.1. Version #1

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k\left(\hat{\theta}_k, c_k\right), a_k \text{ and } c_k \to 0$$

As this is a traditional implementation of SPSA, tuning is accomplished according to Spall [7], namely, $\alpha = 0.602, \gamma = 0.101, a = 4.22, A = 500$ and $c = 3.07$. Such a tune results in iterates $\hat{\theta}_k$ which move by a magnitude of 0.1 in the early iterations. Based on a statistical analysis of the PDFs governing the stochastic demand on schools in the 23 districts, *c* is chosen to be 3.07 which is the mean of the standard deviations of the "noise" functions of each PDF.

4.3.2. Version #2

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k\left(\hat{\theta}_k, c_k\right), a_k \to 0 \text{ and } c_k = 1$$

Since this implementation of SPSA is accomplished with a constant $c_k = 1$, the relevant tune becomes $\alpha = 0.602, a = 4.22, A = 500$. Again, this tune results in iterates $\hat{\theta}_k$ which move by a magnitude of 0.1 in the early iterations. It should be noted that discrete SPSA versions 1 thru 4 utilize this same tune, therefore facilitating a reasonable comparison between them.

4.4. Discrete SPSA

The two methods of Discrete SPSA alluded to in Section 3.2 were implemented in Matlab. Matlab code implemented the projection to feasible hyperplane algorithm discussed in Section 3.2.3. Good use was made of the Matlab function ORTH which is an Orthogonalization algorithm. ORTH, returns an orthonormal basis which spans the same space as the columns of the matrix argument given it. It was this orthonormal basis that "fueled" the projection algorithm.

4.4.1. Version #1

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k\left(\text{Proj}\left(\hat{\theta}_k\right), c_k\right), a_k \text{ and } c_k \to 0$$

For both Version #1 and 2 of discrete SPSA, we use the same tune as in Section 4.3.1, Version #1 above.

4.4.2. Version #2

$$\hat{\theta}_{k+1} = \text{Proj}\left(\text{ProbMove}\left(\hat{\theta}_k - a_k \hat{g}_k\left(\hat{\theta}_k, c_k\right)\right)\right), a_k \text{ and } c_k \to 0$$

4.4.3. Version #3

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k\left(\text{Proj}\left(\hat{\theta}_k\right), c_k\right), a_k \to 0 \text{ and } c_k = 1$$

For both Version #3 and 4 of discrete SPSA, we use the same tune as in Section 4.3.2, Version #2 above.

4.4.4. Version #4

$$\hat{\theta}_{k+1} = \text{Proj}\left(\text{ProbMove}\left(\hat{\theta}_k - a_k \hat{g}_k\left(\hat{\theta}_k, c_k\right)\right)\right), a_k \to 0 \text{ and } c_k = 1$$

4.4.5. Version #5

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k\left(\text{Proj}\left(\hat{\theta}_k\right), c_k\right), a_k = 0.25 \text{ and } c_k = 1$$

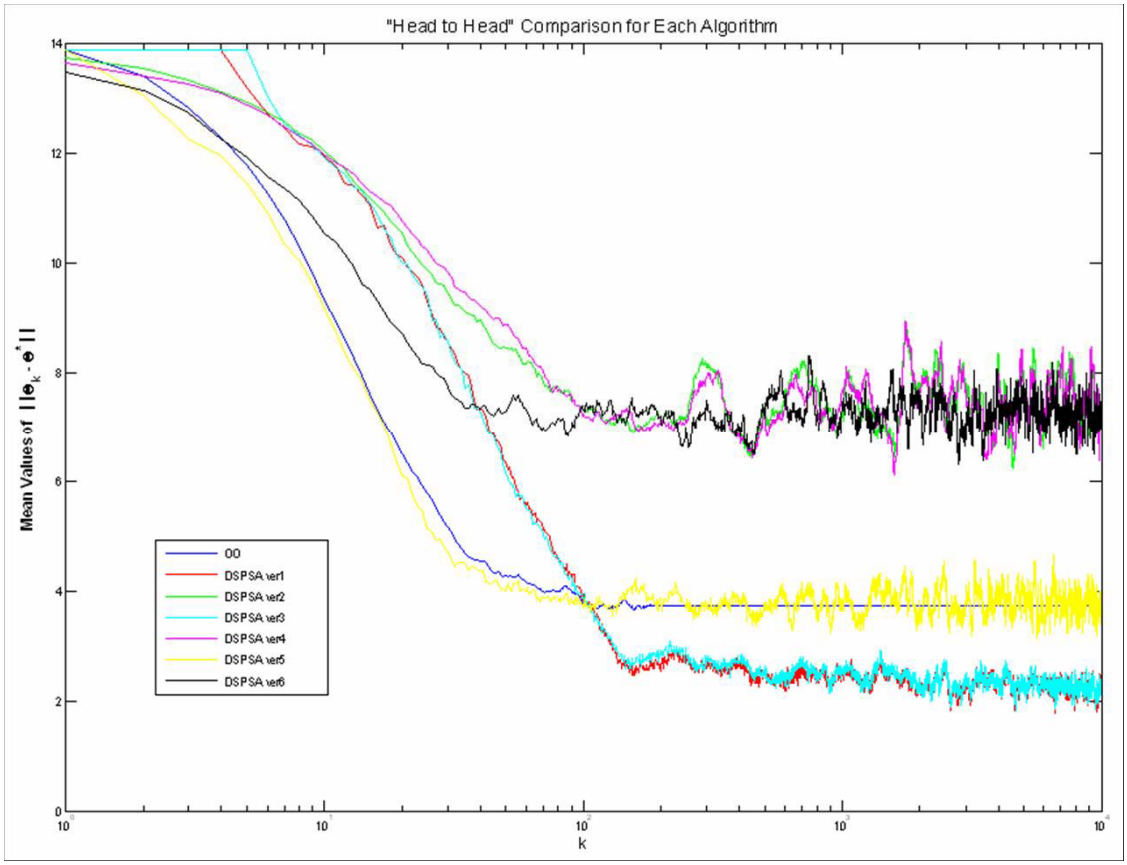
For versions #5 and 6 of DSPSA, we move to a constant gain of $a_k = 0.25$ in addition to the $c_k = 1$.

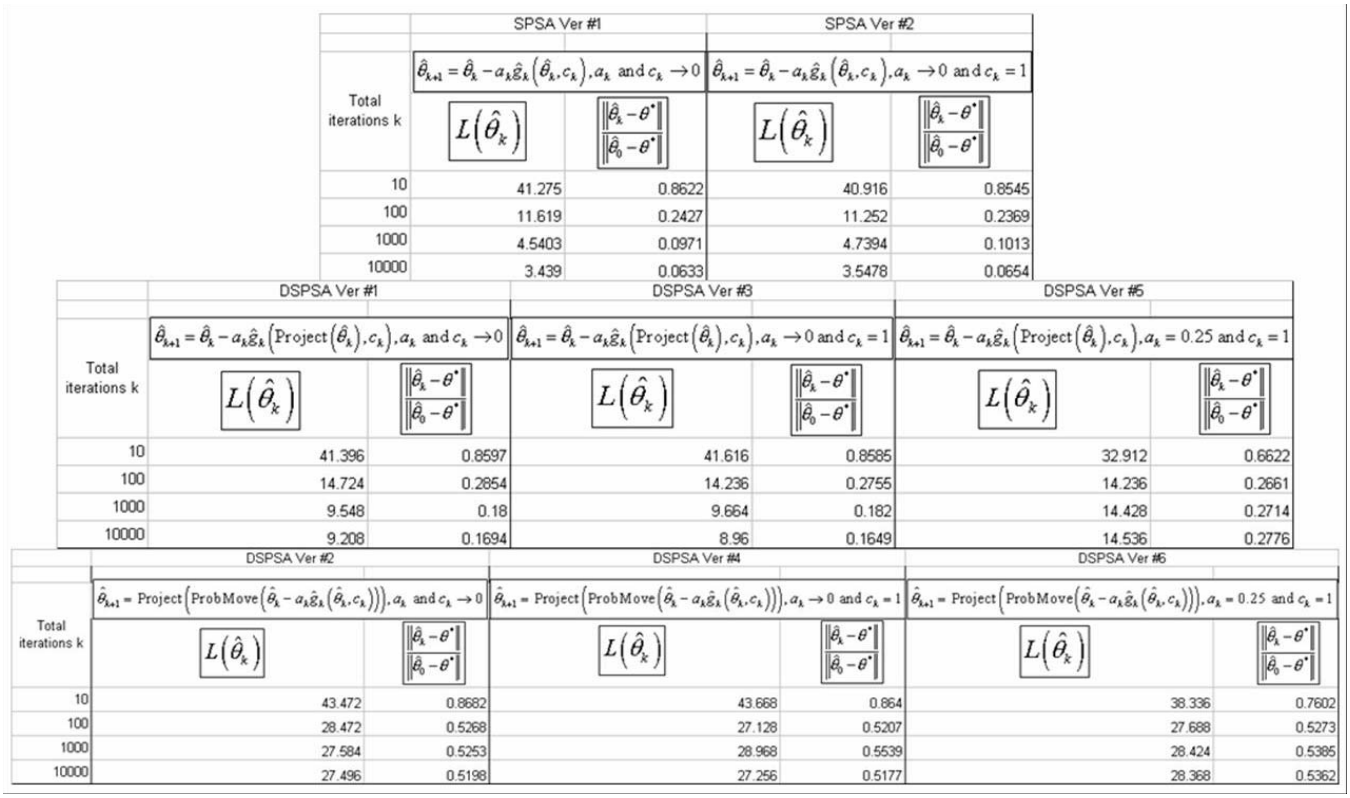
4.4.6. Version #6

$$\hat{\theta}_{k+1} = \text{Proj}\left(\text{ProbMove}\left(\hat{\theta}_k - a_k \hat{g}_k\left(\hat{\theta}_k, c_k\right)\right)\right), a_k = 0.25 \text{ and } c_k = 1$$

5. RESULTS AND DISCUSSION

Discrete SPSA versions #1 and #3 are in general more accurate than Ordinal Optimization. Discrete SPSA version #5 accuracy is on par with Ordinal Optimization. Discrete SPSA versions #2, 4 and 6 are in general less accurate than Ordinal Optimization. Ordinal Optimization and Discrete SPSA version #5 converges to an optimal solution faster than the other form of Discrete SPSA, typically in a little more than 100 iterations.





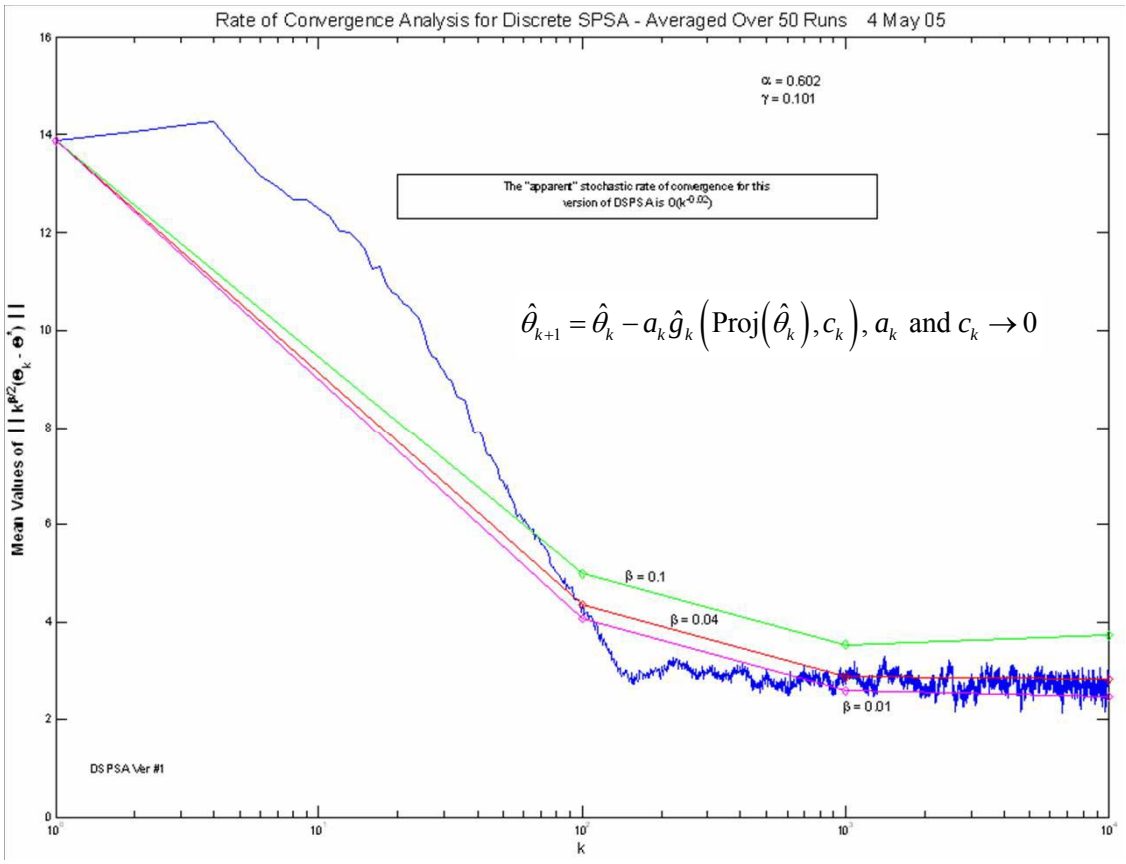
Total iterations k	SPSA Ver #1 $\hat{\theta}_{k+1}=\hat{\theta}_k-a_k\hat{g}_k\left(\hat{\theta}_k,c_k\right),a_k \text{ and } c_k\to 0$ $L\left(\hat{\theta}_k\right)$	SPSA Ver #1 $\frac{\left\|\hat{\theta}_k-\theta^*\right\|}{\left\|\hat{\theta}_0-\theta^*\right\|}$	SPSA Ver #2 $\hat{\theta}_{k+1}=\hat{\theta}_k-a_k\hat{g}_k\left(\hat{\theta}_k,c_k\right),a_k\to 0 \text{ and } c_k=1$ $L\left(\hat{\theta}_k\right)$	SPSA Ver #2 $\frac{\left\|\hat{\theta}_k-\theta^*\right\|}{\left\|\hat{\theta}_0-\theta^*\right\|}$
10	41.275	0.8622	40.916	0.8545
100	11.619	0.2427	11.252	0.2369
1000	4.5403	0.0971	4.7394	0.1013
10000	3.439	0.0633	3.5478	0.0654

Total iterations k	DSPSA Ver #1 $\hat{\theta}_{k+1}=\hat{\theta}_k-a_k\hat{g}_k\left(\text{Project}\left(\hat{\theta}_k\right),c_k\right),a_k \text{ and } c_k\to 0$ $L\left(\hat{\theta}_k\right)$	DSPSA Ver #1 $\frac{\left\|\hat{\theta}_k-\theta^*\right\|}{\left\|\hat{\theta}_0-\theta^*\right\|}$	DSPSA Ver #3 $\hat{\theta}_{k+1}=\hat{\theta}_k-a_k\hat{g}_k\left(\text{Project}\left(\hat{\theta}_k\right),c_k\right),a_k\to 0 \text{ and } c_k=1$ $L\left(\hat{\theta}_k\right)$	DSPSA Ver #3 $\frac{\left\|\hat{\theta}_k-\theta^*\right\|}{\left\|\hat{\theta}_0-\theta^*\right\|}$	DSPSA Ver #5 $\hat{\theta}_{k+1}=\hat{\theta}_k-a_k\hat{g}_k\left(\text{Project}\left(\hat{\theta}_k\right),c_k\right),a_k=0.25 \text{ and } c_k=1$ $L\left(\hat{\theta}_k\right)$	DSPSA Ver #5 $\frac{\left\|\hat{\theta}_k-\theta^*\right\|}{\left\|\hat{\theta}_0-\theta^*\right\|}$
10	41.396	0.8597	41.616	0.8585	32.912	0.6622
100	14.724	0.2854	14.236	0.2755	14.236	0.2661
1000	9.548	0.18	9.664	0.182	14.428	0.2714
10000	9.208	0.1694	8.96	0.1649	14.536	0.2776

Total iterations k	DSPSA Ver #2 $\hat{\theta}_{k+1}=\text{Project}\left(\text{ProbMove}\left(\hat{\theta}_k-a_k\hat{g}_k\left(\hat{\theta}_k,c_k\right)\right)\right),a_k \text{ and } c_k\to 0$ $L\left(\hat{\theta}_k\right)$	DSPSA Ver #2 $\frac{\left\|\hat{\theta}_k-\theta^*\right\|}{\left\|\hat{\theta}_0-\theta^*\right\|}$	DSPSA Ver #4 $\hat{\theta}_{k+1}=\text{Project}\left(\text{ProbMove}\left(\hat{\theta}_k-a_k\hat{g}_k\left(\hat{\theta}_k,c_k\right)\right)\right),a_k\to 0 \text{ and } c_k=1$ $L\left(\hat{\theta}_k\right)$	DSPSA Ver #4 $\frac{\left\|\hat{\theta}_k-\theta^*\right\|}{\left\|\hat{\theta}_0-\theta^*\right\|}$	DSPSA Ver #6 $\hat{\theta}_{k+1}=\text{Project}\left(\text{ProbMove}\left(\hat{\theta}_k-a_k\hat{g}_k\left(\hat{\theta}_k,c_k\right)\right)\right),a_k=0.25 \text{ and } c_k=1$ $L\left(\hat{\theta}_k\right)$	DSPSA Ver #6 $\frac{\left\|\hat{\theta}_k-\theta^*\right\|}{\left\|\hat{\theta}_0-\theta^*\right\|}$
10	43.472	0.8682	43.668	0.864	38.336	0.7602
100	28.472	0.5268	27.128	0.5207	27.688	0.5273
1000	27.584	0.5253	28.968	0.5539	28.424	0.5385
10000	27.496	0.5198	27.256	0.5177	28.368	0.5362

Discrete SPSA versions #1 and #3 behave more like continuous SPSA in terms of algorithm performance, including asymptotic behavior of the iterates projected onto the feasible hyperplane. The rate of convergence analysis for versions #1 and #3 indicates a clear convergence in distribution according to:

$$k^{\beta/2}\left(\hat{\theta}_k-\theta^*\right)\xrightarrow{\text{dist.}}N\left(\mu,\Sigma\right)$$



One can choose a β such that the magnitude of the norms are essentially flat for large *k*, consistent with the above asymptotic normality result. Similarly (and importantly), one can *also* choose a β that will result in the magnitude of the norms decaying to zero. This fact lends some credibility to the empirical estimates of rate of convergence offered in this paper. For instance in Discrete SPSA versions #5 and to a certain extent #6 there could be found no β that would cause the magnitude of the norms to decay to zero. Flatness was indicated whether one chose β to be 0.01, 0.0001 or 0.000001. So which β is the correct apparent speed of convergence? It would seem that the magnitudes of the norms are not really bounded, at least not in the same way as continuous SPSA. And yet Discrete SPSA version #5 performed quite well, as well as Ordinal Optimization. On the average, it converged as quickly as Ordinal Optimization.

The probabilistic movement algorithm of discrete SPSA versions #2, 4 and 6 seems to help keep the SPSA algorithm relatively "well behaved" with excessive noise generally less out at large *k* than in one other implementation. By way of explanation, the author considered yet another instantiation of discrete SPSA that used a "truncate towards minus infinity" function to get the SPSA algorithm to move. The amount of "noise" that was introduced into the algorithm through the "truncate towards minus infinity" function prevented the algorithm from being well behaved asymptotically. In general, the probabilistic movement algorithm performed more reasonably than the "truncate towards minus infinity," hence the reason it was dropped.

Empirically Determined "Apparent" Stochastic Rate of Convergence

DSPSA Ver #1	DSPSA Ver #2	DSPSA Ver #3	DSPSA Ver #4	DSPSA Ver #5	DSPSA Ver #6
$\alpha = 0.602$ $\gamma = 0.101$ $a = 4.22$ $A = 500$ $c = 3.07$	$\alpha = 0.602$ $\gamma = 0.101$ $a = 4.22$ $A = 500$ $c = 3.07$	$\alpha = 0.602$ $a = 4.22$ $A = 500$ $c_k = 1$	$\alpha = 0.602$ $a = 4.22$ $A = 500$ $c_k = 1$	$a_k = 0.25$ $c_k = 1$	$a_k = 0.25$ $c_k = 1$
$O\left(k^{-0.02}\right)$	$O\left(k^{-0.005}\right)$	$O\left(k^{-0.05}\right)$	$O\left(k^{-0.03}\right)$	Not sure if it does	$O\left(k^{-0.0025}\right)$

6. CONCLUSIONS

All algorithms converged into a neighborhood of the optimal solution θ^* with varying degrees of success.

Performance was bounded in this study by the various implementations of Discrete SPSA. One implementation of Discrete SPSA was the most accurate while another implementation of Discrete SPSA was the least accurate. Ordinal Optimization performance figures were right in the middle with one version of Discrete SPSA (one of the constant gain versions) tracking very nicely with the Ordinal Optimization algorithm. This was something of a surprise. Clearly, this version of Discrete SPSA can compete with Ordinal Optimization in terms of speed of convergence versus accuracy.

Through 50 runs of the various DSPSA algorithms, all algorithms converge towards stationary points of the objective function, some quicker and more accurately than others. To the extent that noise seemed minimally invasive, performance was better. One Discrete SPSA algorithm implementation was dropped because the noise evident in the later iterations of *k* rendered the algorithm ill behaved and non-convergent.

Discrete SPSA versions #3 and #4 appeared to converge somewhat faster than versions #1 and #2 in an asymptotic sense, as evidenced by their empirically determined "apparent" stochastic rates of convergence. Version #3 was only marginally better than #1. Despite the fact that versions #5 and #6 employed constant c_k and a_k, convergence seemed dubious. Yet version #5 performed as well as Ordinal Optimization and all the discrete SPSA algorithms seemed to converge in some sense.

7. REFERENCES

[1] Y Ermoliev, "Facility Location Problem," in Numerical Techniques for Stochastic Optimization, Yuri Ermoliev and Roger J-B Wets, eds., Springer, New York, 1988, pp. 413-434.

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

[3] S. D. Hill, L. Gerencser and Z. Vago, "Stochastic Approximation on Discrete Sets Using Simultaneous Perturbation Difference Approximations," Proc. Of the 2003 Conf. On Information Science and Systems, The Johns Hopkins University, March 12-14, 2003.

[4] S. D. Hill, L. Gerencser and Z. Vago, "Stochastic Approximation on Discrete Sets Using Simultaneous Difference Approximations," Proceedings of the 2004 American Control Conference, Boston, Massachusetts, June 30 – July 2, pp. 2795 – 2798.

[5] L. Gerencser, S. D. Hill and Z. Vago, "Optimization Over Discrete Sets via SPSA," Proceedings of the Conference Decision and Control, CDC 38, 1999.

[6] J. C. Spall, S. D. Hill and D. R. Stark, "Theoretical Framework for Comparing Several Stochastic Optimization Approaches," Probabilistic and Randomized Methods for Design under Uncertainty (F. Dabbene and G. Calafiore, eds.), Springer, 2005.

[7] J. C. Spall, Introduction to Stochastic Search and Optimization, Wiley, New Jersey, 2003.

[8] V. Dupac and U. Herkenrath, "Stochastic Approximation on a Discrete Set and the Multi-Armed Bandit Problem," Communications in Statistic – Sequential Analysis, vol. 1, pp. 1-25, 1982.

[9] H. Neuburger, "User Benefit in the Evaluation of Transport and Land Use Plans," in the Journal of Transport Economics and Policy, January, 1971.

[10] S. D. Hill, "Discrete Stochastic Approximation with Application to Resource Allocaton," in the Johns Hopkins APL Technical Digest, January – March, 2005, Volume 26, Number 1, pp. 15 – 21.

Biography

Otis Brooks, a Mission Analyst in the Johns Hopkins University Applied Physics Laboratory (APL) Precision Engagement Business Area, is a member of the Senior Professional Staff, specializing in Modeling & Simulation. He received a BS in Electrical Engineering and a BA in Computer Science from the University of Texas in 1981 and a MS in Applied and Computational Mathematics from The Johns Hopkins University in 2005. Since joining APL in 2000, he has been involved as a Systems Engineer and Analysis with a number of programs, including the F-35 Lightning II (Joint Strike Fighter) Air System and the Tomahawk Weapon System.