3

Theoretical Framework for Comparing Several Stochastic Optimization Approaches

James C. Spall, Stacy D. Hill, and David R. Stark

The Johns Hopkins University, Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, Maryland 20723-6099 USA
{james.spall,stacy.hill,david.stark}@jhuapl.edu

Summary. In this chapter, we establish a framework for formal comparisons of several leading optimization algorithms, providing guidance to practitioners for when to use or not use a particular method. The focus in this chapter is five general algorithm forms: random search, simultaneous perturbation stochastic approximation, simulated annealing, evolution strategies, and genetic algorithms. We summarize the available theoretical results on rates of convergence for the five algorithm forms and then use the theoretical results to draw some preliminary conclusions on the relative efficiency. Our aim is to sort out some of the competing claims of efficiency and to suggest a structure for comparison that is more general and transferable than the usual problem-specific numerical studies.

3.1 Introduction

To address the shortcomings of classical deterministic algorithms, a number of powerful optimization algorithms with embedded randomness have been developed. The population-based methods of evolutionary computation, for example, are one class among many of the available *stochastic* optimization algorithms. Hence, a user facing a challenging optimization problem for which a stochastic optimization method is appropriate meets the daunting task of determining which algorithm is appropriate for a given problem. This choice is made more difficult by some dubious claims that have been made about some popular algorithms. An inappropriate approach may lead to a large waste of resources, both from the view of wasted efforts in implementation and from the view of the resulting suboptimal solution to the optimization problem of interest.

Hence, there is a need for objective analysis of the relative merits and shortcomings of leading approaches to stochastic optimization. This need has certainly been recognized by others, as illustrated, for example, in recent conferences on evolutionary computation, where numerous sessions are devoted to comparing algorithms. Nevertheless, virtually all comparisons have been numerical tests on specific problems. For example, a large fraction of the book [323] is devoted to numerical comparisons. Although sometimes enlightening, such comparisons are severely limited in the *general* insight they provide. Some comparisons for *noisy* evaluations of a simple spherical loss function are given in [15], Chapter 6; however, some of the competitors were implemented in non-standard forms, making the results difficult to interpret for an analyst using a more conventional implementation. Spall [341] also has a number of comparisons (theoretical and numerical) for the cases of noise-free and noisy loss evaluations. At the other end of the spectrum are the 'no free lunch' theorems, [399], which simultaneously consider all possible loss functions and thereby draw conclusions that have limited practical utility since one always has at least *some* knowledge of the nature of the loss function being minimized.

Our aim in this chapter is to lay a framework for a *theoretical* comparison of efficiency applicable to a broad class of practical problems where some (incomplete) knowledge is available about the nature of the loss function. We will consider five basic algorithm forms: random search, simultaneous perturbation stochastic approximation (SPSA), simulated annealing (SAN), and two forms of evolutionary computation (evolution strategy and genetic algorithms). The basic optimization problem corresponds to finding an optimal point θ^*:

$$\theta^* = \arg\min_{\theta \in \Theta} L(\theta),$$

where $L(\theta)$ is the loss function to be minimized, Θ is the domain over which the search will occur, and θ is a p-dimensional vector of parameters. We are mainly interested in the case where θ^* is a *unique* global minimum.

Although stochastic optimization approaches other than the five above exist, we are restricting ourselves to the five general forms in order to be able to make tangible progress (note that there are various specific implementations of each of these general algorithm forms). These five algorithms are general-purpose optimizers with powerful capabilities for serious multivariate optimization problems. Further, they have in common the requirement that they only need measurements of the objective function, not requiring derivative information (gradient or Hessian) for the loss function. It is the long-term expectation that this theoretical framework will provide guidance to those faced with an optimization problem and the associated difficult choice of selecting a suitable method. It is critical to make an informed choice *prior* to investing the considerable resources required given the inherent difficulties in implementing a particular algorithm in a large-scale practical problem (software implementation, data preparation, algorithm tuning, *etc.*).

Central to the approach of this contribution will be the known theoretical analysis on the rate of convergence of each of the candidate algorithms. Our approach will be built as much as possible on *existing* theory characterizing the rates of convergence for the algorithms to perform the comparative analysis. There appears to be no previous analysis putting the theoretical results on a common basis for performing an objective comparison. Of course, this approach has limitations in general because many algorithms have little – or possibly no – theoretical justification. Nonetheless, it is our expectation that performing a formal theoretical comparison of the chosen algorithms will shed light on relative performance of other similar algorithms as well, even if the similar algorithms lack the same current level of theoretical justification.

One might ask whether questions of relative efficiency are relevant in light of the 'no free lunch (NFL)' theorems of [399] and others. The NFL theorems state, in essence, that the expected performance of any pair of optimization algorithms across all possible problems is identical. In practice, of course, one is not interested in solving 'all possible problems,' as there is usually some prior information about the problems of interest and this prior information will affect the algorithm implementation. Hence, the NFL results may not adequately reflect the performance of candidate algorithms as they are actually applied. In other words, some algorithms *do* work better than others on problems of interest. Nevertheless, the NFL results are an important backdrop against which to view the results here, providing limits on the extent to which one algorithm can be claimed as 'better' than another.

In Sections 3.2 through 3.5, we discuss the known convergence rate results on the five algorithm forms under consideration. Section 3.6 then uses these results to provide a theoretical framework for comparison. We demonstrate these results in analyzing the relative efficiency as the problem dimension increases.

3.2 Simple Global Random Search

We first establish a rate of convergence result for the simplest ('blind') random search method where we repeatedly sample over the domain of interest, $\Theta \subseteq \mathbb{R}^p$. This can be done in recursive form or in 'batch' (non-recursive) form by simply laying down a number of points in Θ and taking as our estimate of θ^* that value of θ yielding the lowest L value. A recursive implementation of this idea is as follows.

Step 0 (Initialization). Pick an initial value of θ, say $\hat{\theta}_0$, according to prior information or some probability distribution on the domain Θ. Calculate $L(\hat{\theta}_0)$. Set $k = 0$.

Step 1. Generate a new independent value of θ, say $\theta_{\text{new}}(k)$, according to the chosen probability distribution. If $L(\theta_{\text{new}}(k)) < L(\hat{\theta}_k)$, set $\hat{\theta}_{k+1} = \theta_{\text{new}}(k)$. Else take $\hat{\theta}_{k+1} = \hat{\theta}_k$.

Step 2. Repeat Step 1 until the maximum allowable number of L evaluations has been reached or the user is otherwise satisfied with the current estimate of θ^*.

It is well known that the random search algorithm above will converge in some stochastic sense under modest conditions (see, *e.g.*, [338]). A typical convergence theorem is of the following form (proof in [341], Section 2.2).

Theorem 1. *Suppose that θ^* is the unique minimizer of L on the domain Θ in the sense that $L(\theta^*) = \inf_{\theta\in\Theta} L(\theta)$ and $\inf\{L(\theta) : \|\theta - \theta^*\| \geq \varepsilon\} > L(\theta^*) > -\infty$ for all $\varepsilon > 0$. Suppose further that for any $\varepsilon > 0$ and $\forall k$, there exists a $\delta(\varepsilon) > 0$ such that*

$$\mathbb{P}\{\theta_{\text{new}}(k) : L(\theta_{\text{new}}(k)) < L(\theta^*) + \varepsilon\} \geq \delta(\varepsilon).$$

Then, for the random search algorithm, $\hat{\theta}_k \to \theta^*$ a.s. (almost surely) as $k \to \infty$.

While the above theorem establishes convergence of the simple random search algorithm, it is also of interest to examine the *rate* of convergence. The rate is intended to tell the analyst how close $\hat{\theta}_k$ is likely to be to θ^* for a given cost of search. The cost of search here will be expressed in terms of number of loss function evaluations. Knowledge of the rate is critical in practical applications as simply knowing that an algorithm will eventually converge begs the question of whether the algorithm will yield a practically acceptable solution in any reasonable period. To evaluate the rate, let us specify a 'satisfactory region' $S(\theta^*)$ representing some neighborhood of θ^* providing acceptable accuracy in our solution (*e.g.*, $S(\theta^*)$ might represent a hypercube about θ^* with the length of each side representing a tolerable error in each coordinate of θ). An expression related to the rate of convergence of the above simple random search algorithm is then given by

$$\mathbb{P}\{\hat{\theta}_k \in S(\theta^*)\} = 1 - (1 - \mathbb{P}\{\theta_{\text{new}}(k) \in S(\theta^*)\})^k \tag{3.1}$$

We will use this expression in Section 3.6 to derive a convenient formula for comparison of efficiency with other algorithms.

3.3 Simultaneous Perturbation Stochastic Approximation

The next algorithm we consider is SPSA. This algorithm is designed for continuous variable optimization problems. Unlike the other algorithms here, SPSA is fundamentally oriented to the case of *noisy* function measurements and most of the theory is in that framework. This will make for a difficult comparison with the other algorithms, but Section 3.6 will attempt a comparison nonetheless. The SPSA algorithm works by iterating from an initial guess of the optimal θ, where the iteration process depends on a highly efficient 'simultaneous perturbation' approximation to the gradient $g(\theta) \equiv \partial L(\theta)/\partial\theta$.

Assume that measurements $y(\theta)$ of the loss function are available at any value of θ:

$$y(\theta) = L(\theta) + noise.$$

For example, in a Monte Carlo simulation-based optimization context, $L(\theta)$ may represent the mean response with input parameters θ, and $y(\theta)$ may represent the outcome of one simulation experiment at θ. In some problems, exact loss function measurements will be available; this corresponds to the $noise = 0$ setting (and in the simulation example, would correspond to a deterministic, non-Monte Carlo, simulation). Note that no direct measurements (with or without noise) of the gradient of $L(\theta)$ are assumed available.

It is assumed that $L(\theta)$ is a differentiable function of θ and that the minimum point θ^* corresponds to a zero point of the gradient, *i.e.*,

$$g(\theta^*) \;=\; \left.\frac{\partial L(\theta)}{\partial \theta}\right|_{\theta=\theta^*} \;=\; 0. \tag{3.2}$$

In cases where more than one point satisfies (3.2), there exists theory that ensures that the algorithm will converge to the global minimum, [220]. (As a consequence of the basic recursive form of the algorithm there is generally not a risk of converging to a maximum or saddlepoint of $L(\theta)$, *i.e.*, to non-minimum points where $g(\theta)$ may equal zero.) Another extension of SPSA to global optimization is discussed in [88]. The SPSA procedure has the general recursive SA form:

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

where $\hat{g}_k(\hat{\theta}_k)$ is the estimate of the gradient $g(\theta)$ at the iterate $\hat{\theta}_k$ based on the above-mentioned measurements of the loss function and $a_k > 0$ is a 'gain' sequence. This iterate can be shown to converge under reasonable conditions (*e.g.*, [341] Section 7.3, and [112] for local convergence; [220] for global convergence). The core gradient approximation is

$$\hat{g}_k\left(\hat{\theta}_k\right) \;=\; \frac{y\left(\hat{\theta}_k \;+\; c_k \Delta_k\right) \;-\; y\left(\hat{\theta}_k \;-\; c_k \Delta_k\right)}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{bmatrix}, \tag{3.3}$$

where c_k is some 'small' positive number and the user-generated p-dimensional random perturbation vector, $\Delta_k = [\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kp}]^T$, contains $\{\Delta_{ki}\}$ that are independent and symmetrically distributed about 0 with finite inverse moments $\mathbb{E}(|\Delta_{ki}|^{-1})$ for all k, i. One particular distribution for Δ_{ki} that satisfies these conditions is the symmetric Bernoulli ± 1 distribution; two common distributions that do *not* satisfy the conditions (in particular, the critical finite inverse moment condition) are uniform and normal. The essential basis for efficiency of SPSA in multivariate problems is apparent in (3.3), where only two measurements of the loss function are needed to estimate the p-dimensional gradient vector for any p; this contrasts with the standard finite difference method of gradient approximation, which requires $2p$ measurements.

Most relevant to the comparative analysis goals of this chapter is the asymptotic distribution of the iterate. This was derived in [339], with further developments in [88,112,340]. Essentially, it is known that under appropriate conditions,

$$k^{\beta/2}(\hat{\theta}_k - \theta^*) \xrightarrow{\text{dist}} \mathcal{N}(\mu, \Sigma) \text{ as } k \to \infty, \tag{3.4}$$

where $\beta > 0$ depends on the choice of gain sequences (a_k and c_k), μ depends on both the Hessian and the third derivatives of $L(\theta)$ at θ^* (note that in general, $\mu \neq 0$ in contrast to many well-known asymptotic normality results in estimation), and Σ depends on the Hessian matrix at θ^* and the variance of the noise in the loss measurements. Given the restrictions on the gain sequences to ensure convergence and asymptotic normality, the fastest allowable value for the rate of convergence of $\hat{\theta}_k$ to θ^* is $k^{-1/3}$. This contrasts with the fastest allowable rate of $k^{-1/2}$ for gradient-based algorithms such as Robbins-Monro SA.

Unfortunately, (3.4) is not directly usable in our comparative studies here since the other algorithms being considered here appear to have formal results for convergence rates only for the case of *noise-free* loss measurements. The authors are unaware of any general asymptotic distribution result for the noise-free case (note that it is *not* appropriate to simply let the noise level go to zero in (3.4) in deriving a result for the noise-free case; it is likely that the rate factor β will also change if an asymptotic distribution exists). Some partial results, however, are available that are related to the rate of convergence. Gerencsér [137] established that the moments $\left[\mathbb{E}\left(\left\|\hat{\theta}_k - \theta^*\right\|^q\right)\right]^{1/q}$ converge to zero at a rate of $k^{-1/2}$ for any $q > 0$, when a_k has the standard $1/k$ decay rate. More recently, Gerencsér and Vágó [138] established that the noise-free SPSA algorithm has a geometric rate of convergence when *constant* gains $a_k = a$ are used. In particular, for functions having bounded third derivatives, they show for sufficiently small a,

$$\limsup_{k\to\infty} \frac{\left\|\hat{\theta}_k - \theta^*\right\|}{\eta^k} = 1 \quad \text{a.s.}$$

for some $0 < \eta < 1$. Gerencsér and Vágó [138] go further for quadratic loss functions by specifying η in terms of a and the Hessian matrix of L. Unfortunately, even in the quadratic case, η is not fully specified in terms of quantities associated with L and the algorithm itself (*i.e.*, η depends on unknown constants).

3.4 Simulated Annealing Algorithms

The simulated annealing (SAN) method [187,226] was originally developed for optimization over discrete finite sets. The Metropolis SAN method produces a sequence that converges in probability to the set of global minima of the loss function as T_k, the *temperature*, converges to zero at an appropriate rate.

Gelfand and Mitter [134] present a SAN method for continuous parameter optimization. They obtained discrete-time recursions (which are similar to a stochastic approximation algorithm) for Metropolis-type SAN algorithms that, in the limit, optimize continuous parameter loss functions. Spall ([341] Section 8.6) summarizes this connection of SAN to SA in greater detail. Suppose that $\hat{\theta}_k$ is such a Metropolis-type SAN sequence for optimizing L. To define this sequence, let $q_k(x,\cdot)$ be the p-dimensional Gaussian density function with mean x and variance $b_k^2\sigma_k^2(x)I_p$, where $\sigma_k^2(x) = \max\{1, a_k^\tau\|x\|\}$, τ is fixed in the range $0 < \tau < 1/4$, and $a_k = a/k$ for large k, with $a > 0$. (Observe that $\sup\left\{\sigma_k^2(x), x \in A\right\} \to 1$ as $k \to \infty$ for any bounded set A.) Also, let

$$s_k(x,y) = \begin{cases} \exp\left(-\frac{L(y)-L(x)}{T_k}\right) & \text{if } L(y) > L(x) \\ 1 & \text{otherwise,} \end{cases}$$

where $T_k(x) = b_k^2\sigma_k^2(x)\big/(2a_k)$. The function $s_k(x,y)$ is the *acceptance probability*, as in the usual Metropolis algorithm.

The SAN sequence can be obtained through simulation, in a manner similar to the discrete case:

Step 1. Let $\hat{\theta}_k$ be the current state.
Step 2. Generate a candidate solution $\tilde{\theta}$ according to (the one-step Markov transition) probability density $q_k(\hat{\theta}_k,\cdot)$.
Step 3. Let $\delta_k = L(\tilde{\theta}) - L(\hat{\theta}_k)$. (Then $s_k(\hat{\theta}_k,\tilde{\theta}) \leq 1$, where $s_k(\hat{\theta}_k,\tilde{\theta}) = 1$ if $\delta_k \leq 0$). Let $\hat{\theta}_{k+1} = \tilde{\theta}$ if $\delta_k \leq 0$. Otherwise, consider an independent random variable U_k uniformly distributed on the interval $[0,1]$. Let $\hat{\theta}_{k+1} = \hat{\theta}_k$ if $s_k(\hat{\theta}_k,\tilde{\theta}) > U_k$.

The resulting sequence $\hat{\theta}_k$ has Markov transition probabilities

$$\mathbb{P}\left\{\hat{\theta}_{k+1} \in A \middle| \hat{\theta}_k = x\right\} = \int_A p_k(y|x)dy,$$

where

$$p_k(y|x) = q_k(x,y)s_k(x,y) + r_k(x)\delta(y-x)$$

and $\delta(\cdot)$ is the Dirac-delta function.

Let $\{W_k\}$ be an i.i.d. sequence of p-dimensional standard Gaussian random vectors and let the sequence $\xi_0, \xi_1, \ldots$ be defined by setting

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k(g(\hat{\theta}_k) + \xi_k) + b_kW_k \text{ a.s., } k > 0. \tag{3.5}$$

The reason for introducing this form for the recursion is to show that $\hat{\theta}_k$ converges in probability to the set of global minima of L. This can be shown if we can show that the sequence $\hat{\theta}_k^x$ is tight, where $\hat{\theta}_k^x$ denotes the solution to

(3.5) with initial condition $\hat{\theta}_0 = x$. If $\hat{\theta}_k^x$ is tight, then it can be established that $\hat{\theta}_k^x$ converges in probability, uniformly in x, for x belonging to a compact set K. The limiting distribution is given by the loss function L. In particular, it is the uniform measure on the set of global minima of L. Thus, the main reason for introducing ξ_k is to facilitate the proof of tightness of $\hat{\theta}_k^x$. The sequence $\hat{\theta}_k^x$ is tight under certain restrictions on the sequences a_k and b_k, namely that $a_k = a/k$ (as mentioned above) and $b_k = b/\sqrt{k \log \log k}$ for large k, where a and b are positive constants.

The algorithm is a Metropolis algorithm in the usual sense (*i.e.*, as in the discrete case where the temperature sequence is independent of the state) if almost all $\hat{\theta}_k$ lie in some fixed compact set for all $k > K$, for some $K > 0$, since eventually $\sigma_k^2(\hat{\theta}_k) = 1$. (This assertion follows directly from steps in the proof of Lemma 2(a) in [134], page 121). The sequence $\{\hat{\theta}_k\}$ converges in probability to the global minimum of the loss function. If there is a unique global minimizer θ^*, then the sequence converges in probability to θ^*. To be specific, suppose that $L(\theta)$ has a unique minimum at θ^* and let $S(\theta^*)$ be a neighborhood of θ^*. Gelfand and Mitter [134] show that $\mathbb{P}\{\hat{\theta}_k \in S(\theta^*)\} \to 1$ as $k \to \infty$.

Furthermore, like SPSA, SAN has an asymptotic normality result (but unlike SPSA, this result applies in the noise-free case). In particular, following [403], assume that $a_k = a/k$, $b_k = (b/(k^\gamma \log(k^{1-\gamma} + B_0))^{1/2}$, where B_0, a, and b are positive constants, $0 < \gamma < 1$. Let $H(\theta^*)$ denote the Hessian of $L(\theta)$ evaluated at θ^* and let I_p denote the $p \times p$ identity matrix. Yin [403] showed that

$$[\log(k^{1-\gamma} + B_0)]^{1/2}(\hat{\theta}_k - \theta^*) \to \mathcal{N}(0, \Sigma) \text{ in distribution,}$$

where $\Sigma H + H^T \Sigma + (b/a)I = 0$.

3.5 Evolutionary Computation

3.5.1 General Principles and Theory

Evolutionary computation (EC) represents a class of stochastic search and optimization algorithms that use a Darwinian evolutionary model. The principle feature of an EC algorithm is the search through a population of candidate solutions for the optimal value of a loss function. There are three general approaches in evolutionary computation, namely evolutionary programming (EP), evolution strategies (ES) and genetic algorithms (GA). All three approaches work with a population of candidate solutions and randomly alter the solutions over a sequence of generations according to evolutionary operations of competitive selection, mutation and sometimes recombination (reproduction). The fitness of each population element to survive into the next generation is determined by a selection scheme based on evaluating the loss function for each element of the population. The selection scheme is such that the most favorable elements of the population tend to survive into the next generation while the unfavorable elements tend to perish.

The principal differences in the three approaches are the selection of evolutionary operators used to perform the search and the computer representation of the candidate solutions. EP uses selection and mutation only to generate new solutions. While both ES and GA use selection, recombination and mutation, recombination is used more extensively in GA. A GA traditionally performs evolutionary operations using binary encoding of the solution space, while EP and ES perform the operations using real-coded solutions. The GA also has a real-coded form and there is some indication that the real-coded GA may often be more efficient and provide greater precision than the binary-coded GA ([341], Chapters 9 and 10). The distinction among the three approaches has begun to blur as new hybrid versions of EC algorithms have arisen.

The formal convergence of EC algorithms to the optimal θ^* has been considered in a number of references. Eiben *et al.* [118] derived a convergence in probability result for an elitist GA using the powerful tools of Markov chain theory assuming a finite search space. This result characterized the convergence properties of the GA in terms of the selection, mutation, and recombination probabilities. Rudolph [306] analyzed the basic GA in the binary search space, the canonical GA, without elitist selection. He found that the canonical GA will never converge to the global optimum, and that convergence for this GA comes only by saving the best solution obtained over the course of the search. For function optimization it makes sense to keep the best solution obtained over the course of the search, so convergence is guaranteed. For GA in the binary search space, convergence results that assume a finite search space seem of little practical use; since there are a finite number of points to search, random search and simple enumeration are also guaranteed to converge. However since convergence is a precondition for convergence rate calculations, convergence results assuming a finite search space are not entirely meaningless. Rudolph [309] summarizes the sufficient conditions on the mutation, recombination, and selection probabilities for convergence of EC algorithms in finite search spaces, with a simplified mathematical structure that does not rely on Markov chain theory. Reeves and Rowe [293] and Spall [341], Chapter 10 include a review and further references related to EC convergence theory.

Convergence analysis for EP, ES, and real-valued GA often relies on the Borel-Cantelli Lemma (see for example [21]). The convergence proofs for these algorithms assume that the mutation is applied with non-zero probability such that the joint distribution of new solutions has non-zero probability everywhere. The restrictions made on the mutation operator seem to make these proofs of only academic interest. Convergence properties of EP, ES and real-valued GA may also be derived using the theory of Markov chains. Rudolph [307] details the theory and offers sufficient conditions for convergence of EC algorithms. Other approaches have been taken including modeling EC algorithms as supermartingales as in [307]. Qi and Palmeiri [283] analyzed the real-valued GA assuming an infinite population size. They found that the solutions for a GA using only selection converges in distribution to the distribution concentrated at the global optimum. Also the mean loss value for a real-valued GA with selection and mutation converges to the global optimum. Hart [153] takes a different tack. He defines a class of EC algorithms called evolutionary pattern search algorithms that encompass the real-coded GA, EP, and ES and establishes a stationary point convergence result by extending the convergence theory of generalized pattern search algorithms. The convergence result does not guarantee convergence to the global optimum; it only guarantees that a stationary point is found. Stopping rules related to modifying the mutation probability for the algorithms are provided, however the stopping rules seem to require that the pattern search algorithm structure be adopted.

Global convergence results can be given for a broad class of problems, but the same cannot be said for convergence *rates*. The mathematical complexity of analyzing EC convergence rates is significant. Determining how many generations of the population are required in order to ensure a certain error in the solution is apparently an open problem for arbitrary loss functions. Vose [386,387] showed that assuming an infinite population size, and for every $0 < \delta < 14$, the number of generations required for the GA to come within a distance δ of θ^* is $O(-\log \delta)$. This result is not directly usable in our comparison, however, since it does not give a *quantifiable* expression for the number of generations required to guarantee that the best population element will be within some δ distance of θ^*.

Additional convergence rate results that exist are for restricted classes of loss functions that have some special properties that can be taken advantage of and usually with simplified ECs. In particular, except for the 'big O' result above, [386, 387] (which allows for all three fundamental operations-selection, mutation, and recombination), most of the convergence rate results available are for EC algorithms using selection and mutation only, or using selection and recombination. Both [45] and [307] examine ES algorithms that include selection, mutation and recombination. The function analyzed in both cases is the classic spherical loss function $L(\theta) = \|\theta\|^2$. Convergence rates based on the spherical loss function are somewhat useful, if it is assumed that the sphere approximates a local basin of attraction. A number of other convergence rate results are also available for the spherical loss function; see for example [283] for a real-valued GA.

3.5.2 Convergence Rates for ES Algorithms

This section presents several means by which to determine the rate of convergence for the ES approach to EC. One of the more practically useful convergence rates for EC algorithms applies in a particular class of convex loss functions. The following theorem due to Rudolph [308] is an application of a more general result by Rappl [285]. The theorem is the starting place for the specific convergence rate result that will be used for comparison in Section 3.6.

Definition 1. *An algorithm has a geometric rate of convergence if and only if* $\mathbb{E}[L_k^* - L(\theta^*)] = O(\eta^k)$ *where* $\eta \in (0,1)$ *defines the convergence rate.*

Theorem 2 ([308]). *Let* $\bar{\Theta}_k \equiv \{\hat{\theta}_{k1}, \hat{\theta}_{k2}, \ldots, \hat{\theta}_{kN}\}$ *be the sequence of populations of size N generated by some ES at generation* $k(\hat{\theta}_{ki})$ *represents the* i^{th} *estimate for* θ *from the population of N elements). If* $\mathbb{E}[L_k^* - L(\theta^*)] < \infty$ *and* $\mathbb{E}[L_{k+1}^* - L(\theta^*)|\bar{\Theta}_k] \leq \eta[L_k^* - L(\theta^*)]$ *a.s. for all* $k \geq 0$ *where* $L_k^* = \min\{L(\hat{\theta}_{k1}), L(\hat{\theta}_{k2}), \ldots, L(\hat{\theta}_{kN})\}$*, then the ES algorithm converges a.s. geometrically fast to the optimum of the objective function.*

The condition $\mathbb{E}[L_{k+1}^* - L(\theta^*)|\bar{\Theta}_k] \leq \eta[L_k^* - L(\theta^*)]$ implies that the sequence decreases monotonically on average. This condition is needed since in the $(1,\lambda)$-ES that will be considered below, the loss value of the best parent in the current generation may be worse than the loss value of the best parent of the previous generation, although on average this will not be the case. Rudolph [308] shows that a $(1,\lambda)$-ES using selection and mutation only (where the mutation probability is selected from a uniformly distributed distribution on the unit hyperball), with certain classes of loss functions, satisfies the assumptions of the theorem. One such class is the (K,q)-*strongly convex* functions:

Definition 2. *Let* $L : \Theta \to \mathbb{R}$*. Then L is called* (K,q)*-strongly convex on* Θ *if for all* $x, y \in \Theta$ *and for each* $\alpha \in [0,1]$ *the inequalities*

$$\frac{K}{2}\alpha(1-\alpha)\|x-y\|^2 \leq \alpha L(x) + (1-\alpha)L(y) - L(\alpha x + (1-\alpha)y) \leq \frac{G}{2}\alpha(1-\alpha)\|x-y\|^2$$

hold with $0 < K \leq G \equiv Kq < \infty$.

For example, every quadratic function is (K,q)-strongly convex if the Hessian matrix is positive definite. In the case of twice differentiable functions, fairly simple tests are available for verifying that a function is (K,q)-strongly convex, from Nemirovsky and Yudin [241]. Let ν_1 be the smallest eigenvalue and let ν_2 be the largest eigenvalue of the Hessian matrix. If there exist positive constants K and G such that $0 < K \leq \nu_1 \leq \nu_2 \leq G < \infty$ for all θ then the function L is (K,q)-strongly convex with $q = G/K$. Other tests are possible that only assume the existence of the gradient $g(\theta)$ (see [146]).

The convergence rate result for a $(1,\lambda)$-ES using only selection and mutation on a (K,q)-strongly convex loss function is geometric with a rate of convergence

$$\eta = \left(1 - M_{\lambda,p}^2 q^2\right)$$

where $M_{\lambda,p} = \mathbb{E}[B_{\lambda:\lambda}] > 0$ and where $B_{\lambda:\lambda}$ denotes the maximum of λ independent identically distributed Beta random variables. The computation of

$M_{\lambda,p}$ is complicated since it depends on both the number of offspring λ and the problem dimension p. Asymptotic approximations are available. Assuming p is fixed and $\lambda \to \infty$ then $M_{\lambda,p} \approx (2p^{-1} \log \lambda)^{1/2}$. To extend this convergence rate from a $(1, \lambda)$-ES to a (N, λ)-ES, note that each of the N parents generate λ/N offspring. Then the convergence rate for the (N, λ)-ES where offspring are only obtained by mutation is

$$\eta \leq 1 - \frac{2p^{-1} \log(\lambda/N)}{q^2}$$

for (K, q)-strongly convex functions.

Let us now discuss an alternative method based on approximating the behavior of an idealized (N, λ)-ES as a solution to an ordinary differential equation. Let $r = \left\|\bar{\theta}_k - \theta^*\right\|$, where $\bar{\theta}_k$ is the center of mass (sample mean) of $\{\hat{\theta}_{k1}, \hat{\theta}_{k2}, \ldots, \hat{\theta}_{kN}\}$. Consider a loss function of the spherical-based form $L(\theta) = f\left(\|\theta - \theta^*\|\right)$, where f is a strictly increasing function. Then, an approximate description of the ES is given by the differential equation

$$\frac{dr}{dt} = -\frac{c(t)}{p} r(t),$$

where each time increment (t) of unity represents one iteration of the ES and $c(t)$ is some function dependent on the ES coefficients, [46]. An idealized ES may be based on the assumption that $c(t)$ is a constant, say c^*. As discussed in [46], this is tantamount to knowing the value of r at every time, and normalizing the mutation scale factor at each time so that it is proportional to r. Obviously, this implementation of an ES is idealized because r will almost certainly not be known in practice. Nevertheless, it provides a basis for *some* theoretical analysis. Solving the above differential equation with constant $c(t) = c^*$ and then inverting to solve for t yields

$$t = \frac{p}{c^*} \log\left[\frac{r(0)}{r(t)}\right]. \tag{3.6}$$

Expression (3.6) provides a basis for an estimate of the number of time steps to reach a given distance $r(t)$. Ignoring negligible computation associated with the algorithm itself (*e.g.*, the random number generation), the total cost of the algorithm is then the number of function evaluations per iteration times the number of time steps.

3.5.3 Convergence Rates for GA Algorithms

Based on results in [306] and elsewhere, [341], Section 10.5 and [344] discuss how it is possible to cast the binary bit-based GA in the framework of Markov chains. This allows for a rate of convergence analysis. Consider a GA with a population size of N. Further, suppose that each population element is a binary string of length b bits. Hence, there are 2^b possible strings for an *individual* population element. Then the total number of *unique* possible populations is given by (see [348])

$$N_{\text{pop}} \equiv \frac{(N + 2^b - 1)!}{(2^b - 1)!N!}.$$

It is possible to construct a Markov transition matrix $\mathit{\Pi}$ that provides the probability of transitioning from one population of size N to another population of the same size. This transition matrix has dimension $N_{\text{pop}} \times N_{\text{pop}}$. An individual element in the transition matrix can be computed according to the formulas in [344] (see also [348]). These elements depend in a non-trivial way on the population size, crossover rate, mutation rate, and number of elements considered 'elite.'

Of primary interest in analyzing the performance of GA algorithms using Markov chains is the probability of obtaining a population that contains the optimum θ^*. Let π_k be an $N_{\text{pop}} \times 1$ vector having j^{th} component, $\pi_k(j)$, equal to the probability that the k^{th} generation will result in population j. From basic Markov chain theory,

$$\pi_k^T {=} \pi_{k-1}^T \mathit{\Pi} {=} \pi_0^T \mathit{\Pi}^k$$

where π_0 is an initial probability distribution.

The stationary distribution of the GA is then given by

$$\bar{\pi}^T \equiv \lim_{k\to\infty} \pi_k^T = \lim_{k\to\infty} \pi_0^T \mathit{\Pi}^k.$$

Further, under standard ergodicity assumptions for Markov chains, $\bar{\pi}$ satisfies $\bar{\pi}^T = \bar{\pi}^T \mathit{\Pi}$. This equation provides a mechanism for solving directly for the stationary distribution (*e.g.*, [168], pages 123-124).

Unfortunately, from a practical view, the Markov chain approach has a significant deficiency. The dimension N_{pop} grows very rapidly with increases in the number of bits b and/or the population size N. An estimate of the size of N_{pop} can be obtained by Stirling's approximation as follows:

$$N_{\text{pop}} \approx \sqrt{2\pi} \left(1 + \frac{2^b - 1}{N}\right)^N \left(1 + \frac{N}{2^b - 1}\right)^{2^b - 1} \left(\frac{1}{2^b - 1} + \frac{1}{N}\right)^{1/2}$$

Thus far, our analysis using the above approach has been restricted to scalar θ systems (requiring fewer bits b than a multivariate system) and low N. Examples are given in [341], Section 10.5 and [344]. An approach for compressing the size of the transition matrix (to emphasize only the most likely states) is given in [343]. However, this approach is only useful in an adaptive sense as the algorithm is running; it is not designed for *a-priori* efficiency analysis.

3.6 Comparative Analysis

3.6.1 Problem Statement and Summary of Efficiency Theory for the Five Algorithms

This section uses the specific algorithm results in Sections 3.2 to 3.5 above in drawing conclusions on the relative performance of the five algorithms. There are obviously many ways one can express the rate of convergence, but it is expected that, to the extent they are based on the theory outlined above, the various ways will lead to broadly similar conclusions. We will address the rate of convergence by focusing on the question:

With some high probability $1 - \rho$ (ρ a small number), how many $L(\cdot)$ function evaluations, say n, are needed to achieve a solution lying in some 'satisfactory set' $S(\theta^*)$ containing θ^*?

With the random search algorithm in Section 3.2, we have a closed form solution for use in questions of this sort while with the SPSA, SAN, and EC algorithms of Sections 3.3 through 3.5, we must apply the existing asymptotic results, assuming that they apply to the finite-sample question above. For the GA, there is a finite sample solution using the Markov chain approach. For each of the five algorithms, we will outline below an analytical expression useful in addressing the question. After we have discussed the analytical expressions, we present a comparative analysis in a simple problem setting for varying p.

Random Search

We can use (3.1) to answer the question above. Setting the left-hand side of (3.1) to $1 - \rho$ and supposing that there is a constant sampling probability $P^* = \mathbb{P}\{\theta_{\text{new}}(k) \in S(\theta^*)\}$ for all k, we have

$$n = \frac{\log\ \rho}{\log\ (1 - P^*)}. \tag{3.7}$$

Although (3.7) may appear benign at first glance, this expression grows rapidly as p gets large due to P^* approaching 0. (A numerically stable approximation that is useful with small P^* is given in [341], page 62). Hence, (3.7) shows the extreme inefficiency of simple random search in higher-dimensional problems as illustrated in the study below. Note that while (3.7) is in terms of the iterate $\hat{\theta}_k$, a result related to the rate of convergence for $L(\hat{\theta}_k)$ is given in [265], page 24; this result is in terms of extreme value distributions and also confirms the inefficiency of simple random search algorithms in high-dimensional problems.

Simultaneous Perturbation Stochastic Approximation

As mentioned in Section 3.4, there is no known asymptotic normality result in the case of noise-free measurements of $L(\theta)$ (although Gerencsér and

Vágó, [138], show that the rate of convergence is geometric with an unknown constant governing the decay). Nonetheless, a *conservative* representation of the rate of convergence is available by assuming a noisy case with small levels of noise. Then we know from (4.4) that the approximate distribution of $\hat{\theta}_k$ with optimal decay rates for the gains a_k and c_k is $\mathcal{N}(\theta^* + \mu/k^{1/3}, \Sigma/k^{2/3})$. In principle, then, one can use this distribution to compute the probabilities associated with arbitrary sets $S(\theta^*)$, and these probabilities will be directly a function of k. In practice, due to the correlation in Σ, this may not be easy and so inequalities such as in [363], Chapter 2 can be used to provide bounds on $\mathbb{P}\{\hat{\theta}_k \in S(\theta^*\})$ in terms of the marginal probabilities of the $\hat{\theta}_k$ elements.

For purposes of insight, consider a case where the covariance matrix Σ is diagonal. If $S(\theta^*)$ is a hypercube of the form $[s_1^-, s_1^+] \times [s_2^-, s_2^+] \times ... \times [s_p^-, s_p^+]$, then $\mathbb{P}\{\hat{\theta}_k \in S(\theta^*)\}$ is a product of the marginal normal probabilities associated with each element of $\hat{\theta}_k$ lying in its respective interval $[s_i^-, s_i^+]$, $i = 1, 2, \dots, p$. Such diagonal covariance matrices arise when the loss function is separable in each of the components of θ. Then we can find the k such that the product of probabilities equals $1 - \rho$. To illustrate more specifically, suppose further that $\Sigma = \sigma^2 I$, the $\mu/k^{1/3}$ term in the mean is negligible, that $S(\theta^*)$ is centered around θ^*, and that $\delta s \equiv s_i^+ - s_i^-$ for all i. (*i.e.*, $s_i^+ - s_i^-$ does not depend on i). Then for a specified ρ, we seek the n such that $\mathbb{P}\{\hat{\theta}_k \in S(\theta^*)\} = \mathbb{P}\{\hat{\theta}_{ki} \in [s_i^-, s_i^+]\}^p = 1 - \rho$. From standard $\mathcal{N}(0,1)$ distribution tables, there exists a displacement factor, say $d(p)$, such that the probability contained within $\pm d(p)$ units contains probability amount $(1-\rho)^{1/p}$; we are interested in the k such that $2d(p)\sigma/k^{1/3} = \delta s$. From the fact that SPSA uses two $L(\theta^*)$ evaluations per iteration, the value n to achieve the desired probability for $\hat{\theta}_k \in S(\theta^*)$ is then

$$n = 2\left(\frac{2d(p)\sigma}{\delta s}\right)^3 .$$

Unfortunately, the authors are unaware of any convenient analytical form for determining $d(p)$, which would allow a ‘clean’ analytical comparison with the efficiency formula (3.7) above (a closed-form approximation to normal probabilities of intervals is given in [171], pages 55-57, but this approximation does not yield a closed form for $d(p)$).

Simulated Annealing

Because SAN, like SPSA, has an asymptotic normality result, the method above for characterizing the rate of convergence for SPSA may also be used here. Again, we shall consider the case where the covariance matrix is diagonal ($\Sigma = \sigma^2 I$). Assume also that $S(\theta^*)$ is a hypercube of the form $[s_1^-, s_1^+] \times [s_2^-, s_2^+] \times ... \times [s_p^-, s_p^+]$ centered around θ^*, and that $\delta s \equiv s_i^+ - s_i^-$, for all i. The (positive) constant B_0 is assumed small enough that it can be ignored. At each iteration after the first, SAN must evaluate $L(\theta^*)$ only once per iteration. So the value n to achieve the desired probability for $\hat{\theta}_k \in S(\theta^*)$ is

$$\log n^{1-\gamma} \;=\; \left(\frac{2d(p)\sigma}{\delta s}\right)^2 .$$

Evolution Strategy

As discussed in Section 3.5, the rate-of-convergence results for algorithms of the evolutionary computation type are not as well developed as for the other three algorithms of this chapter. Theorem 2 gives a general bound on $\mathbb{E}[L(\hat{\theta}_k)-L(\theta^*)]$ for application of a (N,λ)-ES form of EC algorithm to (K,q)-strongly convex functions. A more explicit form of the bound is available for the $(1,\lambda)$-ES. Unfortunately, even in the optimistic case of an explicit numerical bound on $\mathbb{E}[L(\hat{\theta}_k)-L(\theta^*)]$, we cannot readily translate the bound into a probability calculation for $\hat{\theta}_k \in S(\theta^*)$, as used above (and, conversely, the asymptotic normality result on $\hat{\theta}_k$ for SPSA and SAN cannot be readily translated into one on $L(\hat{\theta}_k)$ since $\partial L/\partial\theta = 0$ at θ^*, see, *e.g.*, [327], pages 122-124, although Lehmann in [204], pages 338-339 suggests a possible means of coping with this problem via higher-order expansions). So, in order to make *some* reasonable comparison, let us suppose that we can associate a set $S(\theta^*)$ with a given deviation from $L(\theta^*)$, *i.e.*, $S(\theta^*) = \{\theta : L(\hat{\theta}_k) - L(\theta^*) \le \varepsilon\}$ for some prespecified tolerance $\varepsilon > 0$ (note that $S(\theta^*)$ is a function of ε). As presented in [308], $\mathbb{E}[L(\hat{\theta}_k)-L(\theta)] \le \eta^k$ for sufficiently large k, where η is the convergence rate in Section 3.5. Then by Markov's inequality,

$$1-\mathbb{P}\{\hat{\theta}_k \in S(\theta^*)\} \le \frac{\mathbb{E}[L(\hat{\theta}_k)-L(\theta^*)]}{\varepsilon} \le \frac{\eta^k}{\varepsilon} \tag{3.8}$$

indicating that $\mathbb{P}\{\hat{\theta}_k \in S(\theta^*)\}$ is bounded below by the ES bounds mentioned in Section 3.5. For EC algorithms in general (and ES in particular), there are λ evaluations of the loss function for each generation k so that $n = \lambda k$, where

$$k = \frac{\log\rho - \log(1/\varepsilon)}{\log\left[1-\frac{2}{pq^2}\log(\lambda/N)\right]}. \tag{3.9}$$

We also report results related to the differential equation solution (3.6). As noted, this solution is tied to some restrictions, namely to loss functions of the spherical-based form $L(\theta) = f\left(\|\theta-\theta^*\|\right)$ and to an idealized ES with a mechanism for adaptively scaling the mutation magnitude according to the current distance $r = \left\|\bar{\theta}_k - \theta^*\right\|$. Further, as a deterministic approximation to a stochastic algorithm, there is no simple way to determine the probability ρ defined above. If, as mentioned above, we consider $S(\theta^*)$ in the form of a hypercube $[s_1^-, s_1^+] \times [s_2^-, s_2^+] \times ... \times [s_p^-, s_p^+]$, we can specify a r_{inside} that defines the radius of the largest hypersphere that is contained within the hypercube and r_{outside} that defines the radius of the smallest hypersphere that lies outside (*i.e.*, contains) the hypercube. The number of function evaluations needed to yield a solution in $S(\theta^*)$ is then bounded above and below by the number required for a solution to lie in these inside and outside hyperspheres. That is, substituting r_{inside} or r_{outside} for $r(t)$ in the right-hand side of (3.6) yields an upper and lower bound, respectively, to the number of time steps, which, by the appropriate multiplication, yields bounds to the number of function evaluations.

Genetic Algorithm

As mentioned in Section 3.5, while the GA has a relatively clean theory that applies in both finite and asymptotic samples, there are significant challenges in computing the elements of the Markov transition matrix $\mathit{\Pi}$. The number of possible states – corresponding to the number N_{pop} of possible populations – grows extremely rapidly with the number of population elements N or the number of bits b. The computation of the $N_{\text{pop}} \times N_{\text{pop}}$ transition matrix $\mathit{\Pi}$ quickly overwhelms even the most powerful current or foreseeable personal computers.

Nevertheless, in principle, the Markov structure is convenient for establishing a convergence rate for the GA. Recall that π_k is the $N_{\text{pop}} \times 1$ vector having j^{th} component, $\pi_k(j)$, equal to the probability that the k^{th} generation will result in population j. Let us denote by S_J the set of indices j such that population j contains at least one member lying inside $S(\theta^*)$. Hence, $S_J \subseteq \{1, 2, \ldots, N\}$. Then

$$n = N + (N - N_{\text{elite}}) \min \left\{ k : \sum_{j \in S_J} \pi_k(j) \geq 1 - \rho \right\},$$

where N_{elite} denotes the number of elite elements in the population being saved from one generation to the next and we have assumed that all non-elite function evaluations are not ‘saved’ from one generation to the next (*i.e.*, every generation entails $N - N_{\text{elite}}$ function evaluations).

3.6.2 Application of Convergence Rate Expressions for Varying *p*

We now apply the results above to demonstrate relative efficiency for varying p. Because the GA result is computationally explosive as p gets larger (requiring a larger bit string length and/or population size), we restrict the comparison here to the four algorithms: random search, SPSA, SAN and ES. Let $\mathit{\Theta} = [0, 1]^p$ (the p-dimensional hypercube with minimum and maximum θ values of 0 and 1 for each component). We want to guarantee with probability 0.90 that each element of θ is within 0.04 units of the optimal. Let the (unknown) optimal θ, θ^*, lie in $(0.04, 0.96)^p$. The individual components of θ^* are θ_i^*. Hence,

$$S(\theta^*) = [\theta_1^* - 0.04,\ \theta_1^* + 0.04] \times [\theta_2^* - 0.04,\ \theta_2^* + 0.04] \times \dots$$
$$\times [\theta_p^* - 0.04,\ \theta_p^* + 0.04] \subset \Theta.$$

Table 3.1 is a summary of relative efficiency for the setting above for $p = 2, 5$, and 10; the efficiency is normalized so that all algorithms perform equally at $p = 1$, as described below. The numbers in Table 3.1 are the ratios of the number of loss measurements for the given algorithm over the number for the best algorithm at the specified p; the highlighted values 1.0 indicate the best algorithm for each of the values of p. To establish a fair basis for comparison, we fixed the various parameters in the expressions above (*e.g.*, σ in SPSA and SAN, λ for the ES, *etc.*) so that the algorithms produced identical efficiency results for $p = 1$ (requiring $n = 28$ measurements to achieve the objective outlined above). These parameters do not explicitly depend on p. We then use these parameter settings as p increases. Of course, in practice, algorithm parameters are typically tuned for each new problem, including changes in p. Hence, the results may not reflect practical relative efficiency, including the cost of the tuning process. Rather, they point towards general efficiency trends as a function of problem dimension in the absence of problem-specific tuning.

For the random sampling algorithm, suppose uniform sampling on Θ is used to generate $\theta_{\text{new}}(k)$ for all k. Then, $P^* = 0.08^p$. For SPSA, we fix σ such that the same number of function measurements in the $p = 1$ case ($n = 28$) is used for both random search and SPSA (so $\delta s = 0.08$ and $\sigma = 0.0586$). Likewise, for SAN, we fix σ to achieve the same objective (so $\delta s = 0.08$ and $\sigma = 0.031390$). Also, for convenience, take $\gamma = 1/2$. To compare the (N, λ)-ES algorithm with the random search, SPSA, and SAN algorithms, it is assumed that the loss function is restricted to the (K, q)-strongly convex functions or spherical-based forms discussed in Section 3.5. Also let $\lambda = 14$, $N = 7$, $\varepsilon = 8.3$, $q = 4$, and $\rho = 0.1$. The variables were constrained here so that for $p = 1$, we have the same n ($= 28$) as realized for the other algorithms. Table 3.1 summarizes the performance comparison results.

Table 3.1. Ratios of loss measurements needed relative to best algorithm at each p, for $1 \le p \le 10$

	$p = 1$	$p = 2$	$p = 5$	$p = 10$
Random search	**1.0**	11.6	8970	2×10^9
SPSA	**1.0**	1.5	**1.0**	**1.0**
SAN	**1.0**	**1.0**	2.2	4.1
ES (from (3.8), (3.9))	**1.0**	1.9	1.9	2.8
ES (from (3.6) w. inside hypersphere)	**1.0**	2.1	2.4	3.8
ES (from (3.6) w. outside hypersphere)	**1.0**	1.8	1.8	2.6

Table 3.1 illustrates the explosive growth in the relative (and absolute) number of loss evaluations needed as p increases for the random search algorithm. The other algorithms perform more comparably, but there are still some non-negligible differences. For example, at $p = 5$, SAN will take 2.2 times more loss measurements than SPSA to achieve the objective of having $\hat{\theta}_k$ inside $S(\theta^*)$ with probability 0.90. Of course, as p increases, all algorithms take more measurements; the table only shows *relative* numbers of function evaluations (considered more reliable than absolute numbers).

This large improvement of SPSA and SAN relative to random search may partly result from the more restrictive regularity conditions of SPSA and SAN (*i.e.*, for formal convergence, SPSA assumes a several-times-differentiable loss function) and partly from the fact that SPSA and SAN work with *implicit* gradient information via gradient approximations. (The reasons for improvement with ES are less clear due to the lack of an identifiable connection to the gradient.) Of course, to maintain a fair comparison, SPSA and SAN, like the other algorithms here, explicitly use only loss evaluations, no direct gradient information. On the other hand, there are some differences between SPSA and SAN. The different gradient approximations in SPSA and SAN may explain their relative efficiency. The 'Metropolis-type approximation appears to be much farther away from an exact gradient-based algorithm than a finite-difference approximation' ([134], page 128). By contrast, SPSA, recall, uses a (highly efficient) finite-difference-like approximation to the gradient.

The performance for ES is quite good. The restriction to strongly convex loss functions (from (3.8) and (3.9)) or spherical losses (from (3.6)), however, gives the ES in this setting a strong structure not available to the other algorithms. It remains unclear what practical theoretical conclusions can be drawn on a broader class of problems. More advanced sensitivity studies for various λ, N, and q have not yet been completed. Further, the inequality in (3.8) provides an optimistic assessment of the convergence rate. Ideally, a more general rate-of-convergence theory will provide a more broadly applicable basis for comparison.