Random Search Algorithms

Zelda B. Zabinsky*

April 5, 2009

Abstract

Random search algorithms are useful for many ill-structured global optimization problems with continuous and/or discrete variables. Typically random search algorithms sacrifice a guarantee of optimality for finding a good solution quickly with convergence results in probability. Random search algorithms include simulated annealing, tabu search, genetic algorithms, evolutionary programming, particle swarm optimization, ant colony optimization, cross-entropy, stochastic approximation, multi-start and clustering algorithms, to name a few. They may be categorized as global (exploration) versus local (exploitation) search, or instance-based versus model-based. However, one feature these methods share is the use of probability in determining their iterative procedures. This article provides an overview of these random search algorithms, with a probabilistic view that ties them together.

A random search algorithm refers to an algorithm that uses some kind of randomness or probability (typically in the form of a pseudo-random number generator) in the definition of the method, and in the literature, may be called a Monte Carlo method or a stochastic algorithm. The term metaheuristic is also commonly associated with random search algorithms. Simulated annealing, tabu search, genetic algorithms, evolutionary programming, particle swarm optimization, ant colony optimization, cross-entropy, stochastic approximation, multi-start, clustering algorithms, and other random search methods are being widely applied to continuous and discrete global optimization problems, see, for example, [7, 8, 24, 45, 47, 64, 67, 73]. Random search algorithms are useful for ill-structured global optimization problems, where the objective function may be nonconvex, nondifferentiable, and possibly discontinuous over a continuous, discrete, or mixed continuous-discrete domain. A global optimization problem with continuous variables may contain several local optima or stationary points. A problem with discrete variables falls into the category of combinatorial optimization and is often typified by the Traveling Salesperson Problem (TSP). A combination of continuous and discrete variables arises in many complex systems including engineering design problems, scheduling and sequencing problems, and other applications in biological and economic systems.

The problem of designing algorithms that obtain global optimal solutions is very difficult when there is no overriding structure that indicates whether a local solution is indeed a global

*Department of Industrial and Systems Engineering, University of Washington, Seattle, WA, 98195–2650, USA, zelda@u.washington.edu

solution. In contrast to deterministic methods (such as branch and bound, interval analysis, and tunnelling methods [27, 28, 45]), which typically guarantee asymptotic convergence to the optimum, random search algorithms ensure convergence in probability. The tradeoff is in terms of computational effort. Random search algorithms are popular because they can provide a relatively good solution quickly and easily.

Random search methods have been shown to have a potential to solve large-scale problems efficiently in a way that is not possible for deterministic algorithms. Whereas it is known that a deterministic method for global optimization is NP-hard [69], there is evidence that a stochastic algorithm can be executed in polynomial time, on the average. For instance, Dyer and Frieze [21, 22] showed that estimating the volume of a convex body takes an exponential number of function evaluations for *any* deterministic algorithm, but if one is willing to accept a weaker claim of being correct with an estimate that has a high probability of being correct, then a stochastic algorithm can provide such an estimate in polynomial time.

Another advantage of random search methods is that they are relatively easy to implement on complex problems with "black-box" function evaluations. Because the methods typically only rely on function evaluations, rather than gradient and Hessian information, they can be coded quickly, and applied to a broad class of global optimization problems. A disadvantage of these methods is that they are currently customized to each specific problem largely through trial and error. A common experience is that random search algorithms perform well and are "robust" in the sense that they give useful information quickly for ill-structured global optimization problems.

Different types of categorizations have been suggested for random search algorithms. Schoen [57] provides a classification of a two-phase method, where one phase is a global search phase and the other is a local search phase. An example of a two-phase method is multi-start, where a local search algorithm is initiated from a starting point that is globally, typically uniformly, distributed. Often the local phase in multi-start is a deterministic gradient search algorithm, although researchers have experimented with using simulated annealing in the local phase of multi-start. The global phase can be viewed as an exploration phase aimed at exploring the entire feasible region, while the local phase can be viewed as an exploitation phase aimed at exploiting local information (e.g. gradient or nearest neighbor information).

Zlochin [77] categorizes random search algorithms as instance-based or model-based, where instance-based methods generate new candidate points based on the current point or population of points, and model-based methods rely on an explicit sampling distribution and update parameters of the probability distribution. Examples of instance-based algorithms include simulated annealing, genetic algorithms, and tabu search, whereas examples of model-based algorithms include ant colony optimization, stochastic gradient search, and the cross-entropy method.

To understand the differences between random search algorithms, we concentrate on the procedures of generating new candidate points, and updating the collection of points maintained. The next section describes generating and updating procedures for several random search algorithms. Even though instance-based methods may use heuristics to generate and update points, they do implicitly induce a sampling distribution. Abstracting the random search algorithms by a sequence of sampling distributions (whether implicit or explicit) provides a means to understand the performance of the algorithms. In Section 2 we analyze the performance of Pure Random Search, Pure Adaptive Search and Annealing Adaptive

Search by investigating their sampling distributions. We discuss model-based methods in the context of their explicit sampling distribution in Section 3. We conclude in Section 4 with a meta-control framework to adapt algorithm parameters affecting sampling distributions. We hope this overview serves to assist researchers in using random search algorithms and developing better methods in the future.

1 Generic Random Search and Specific Algorithms

The general global optimization problem (P) used here is defined as,

$$(P) \qquad \min_{x \in S} f(x)$$

where x is a vector of n decision variables, S is an n-dimensional feasible region and assumed to be nonempty, and f is a real-valued function defined over S. The goal is to find a value for x contained in S that minimizes f. The objective function may be an oracle, or black-box function, in the sense that an explicit mathematical expression is not required but a numerical value for f must be returned for an input value of x. The feasible set S may be represented by a system of nonlinear equations, such as $g_j(x) \leq 0$ for $j = 1, \ldots, m$, or by a membership function that simply returns whether or not a given x is in S. Notice that the feasible region may include both continuous and discrete variables. Denote the global optimal solution to (P) by (x_*, y_*) where

$$x_* = \arg\min_{x \in S} f(x) \quad \text{and} \quad y_* = f(x_*) = \min_{x \in S} f(x).$$

Random search algorithms are also frequently applied to stochastic optimization problems (also known as simulation-optimization) where the objective function and constraints involve randomness, but in this article we assume that the objective and membership functions in (P) are deterministic.

In order to ensure a global optimum exists, we need to assume some regularity conditions. If (P) is a continuous problem and the feasible set S is nonempty and compact, then the Weierstrass theorem from classical analysis guarantees the existence of a global solution [39]. If (P) is a discrete problem and the feasible set S is nonempty and finite, a global solution exists. The existence of a global optimum can be guaranteed under slightly more general conditions. Note that the existence of a *unique* minimum at x_* is not required. If there are multiple optimal minima, let x_* be an arbitrary fixed global minimum.

In order to describe random search algorithms on continuous and/or discrete domains, it is necessary to assume some concept of neighborhood. In Euclidean space (real or integer valued variables), the notion of neighborhood is typically defined by a ball of radius δ, $\delta > 0$, of points that are within a (Euclidean) distance δ of a point $x \in S$. However, the distance between points could be associated with the algorithm, rather than the definition of the problem. For example, the TSP in combinatorial optimization has several possible neighborhood structures (e.g. one-city swap, two-city swap) that are integrally associated with the search algorithm rather than the statement of the problem. Even for problems with a continuous domain, the neighborhood associated with an algorithm that samples within a

fixed radius of its current point is local as compared to an algorithm that samples broadly over the entire feasible region.

We describe a generic random search algorithm by a sequence of iterates $\{X_k\}$ on iteration $k = 0, 1, \ldots$ which may depend on previous points and algorithmic parameters. The current iterate X_k may represent a single point, or a collection of points, to include population-based algorithms. The iterates are also capitalized to denote that they are random variables, reflecting the probabilistic nature of the random search algorithm.

Generic Random Search Algorithm

Step 0. Initialize algorithm parameters Θ_0, initial points $X_0 \subset S$ and iteration index $k = 0$.

Step 1. Generate a collection of candidate points $V_{k+1} \subset S$ according to a specific *generator* and associated *sampling distribution*.

Step 2. Update X_{k+1} based on the candidate points V_{k+1}, previous iterates and algorithmic parameters. Also update algorithm parameters Θ_{k+1}.

Step 3. If a stopping criterion is met, stop. Otherwise increment k and return to Step 1.

This generic random search algorithm depends on two basic procedures, the generator in Step 1 that produces candidate points, and the update procedure in Step 2. We first discuss some examples for the generator procedure.

Single-point Generators Many random search algorithms maintain and generate a single point at each iteration. For these single-point generators, the candidate point V_{k+1} is generated based on a combination of the current point and previous points. A common method to generate a candidate point is to take a step size in a vector direction, called a direction-step paradigm in [47], or also known as step size algorithms. Step 1 of a step size algorithm can be expressed by

$$V_{k+1} = X_k + S_k D_k$$

where the candidate point is generated by taking a step from the current point X_k of length S_k in a specified direction D_k on iteration k. In continuous problems, the direction of movement D_k may be motivated by gradient information, and the step length may be the result of a line search. Quasi-Newton methods take advantage of an approximation of the Hessian to provide a search direction. In the basic method of stochastic approximation, also referred to as the Robbins-Monroe algorithm, the step length is referred to as the "gain" (see [64] for a good discussion). If the step direction is closely related to the gradient, as in stochastic gradient search, then local convergence may be proven under certain conditions, however other variations must be introduced to escape local minima and find the global minimum.

As an alternative, a direction D_k that does not use any local information and is generated according to a uniform distribution on a hypersphere has been investigated. A collection of step size algorithms, including [44, 48, 49, 58, 59, 63, 75], obtain a direction vector by sampling from a uniform distribution on a unit hypersphere. The method of choosing the step size may also be randomly generated, and may shrink or expand depending on the

success of previously sampled points. It is interesting that several of these step size random search algorithms report experiencing computation that is linear in dimension.

The algorithm Improving Hit-and-Run (IHR) also generates a direction uniformly on a hypersphere, and generates a step size according to a uniform distribution on the feasible portion of the line segment in direction D_k [75]. This simple generator is motivated by convergence properties of a Markov chain Monte Carlo sampler called Hit-and-Run to a target distribution [5, 6, 38, 62]. The Hit-and-Run generator coupled with an update procedure has been used in several simulated annealing algorithms [52, 53, 54]. The update procedure in Improving Hit-and-Run is simply to only accept improving points. Even though this is a straightforward random search algorithm, its expected performance on a class of spherical programs is polynomial $O(n^{5/2})$ in dimension [75].

Examples of generators on discrete domains include nearest neighbor or local search, ball walk, k-city swap for TSP, and more recently, Very Large-Scale Neighborhood Search (VLSN) [2]. Classical Markov chains such as the nearest neighbor random walk or the coordinate direction random walk can get trapped in isolated regions of the domain. A variation of Hit-and-Run has been proven to converge to an arbitrary target distribution over an arbitrary subset S of an integer hyper-rectangle [4]. An extension of Hit-and-Run to mixed continuous/integer domains using a pattern generator has recently been embedded in IHR for global optimization [40]. The generator in tabu search is known for its restriction on neighborhoods - it uses a local neighborhood but prevents points from being generated that have been recently visited [24, 25].

Multiple-point Generators Population-based random search algorithms use a collection of current points to generate another collection of candidate points. Many of these algorithms are motivated by biological processes, and include genetic algorithms, evolutionary programming, particle swarm optimization and ant colony optimization. The concept behind genetic algorithms and evolutionary programming [3, 13, 33] is to produce "off-spring" of the current population with parameters related to reproduction (e.g. genetic cross-over). Particle swarm optimization [30, 31] is based on the social behavior of birds and schools of fish, communicating to find food. Once an individual in the group finds a food source (associated with an objective function value), it informs the others. While initially the individuals of the group behave independently, they eventually communicate and move collectively towards one of the found sources of food. The individuals, or particles in the swarm, are typically modeled by position and velocity, which are updated based on the success of the other particles in the group. Ant colony optimization [20] is also considered a swarm intelligence, where the population of ants are moving individually, but communicate through a pheromone that provides feedback as to successful locations.

Another multiple-point generator that uses a population of points is differential evolution [66] which uses a weighted difference between two population vectors to generate a third vector. It is similar to the particle swarm generator because no explicit probability distribution is used, but an implicit empirical distribution is created through the interaction of the points. The generator procedure is just telling part of the story. The update procedure also influences the resulting probability distribution underlying the random search method.

Update Procedure After a candidate point is generated, Step 2 of the generic random search algorithm specifies a procedure to update the current point and algorithm parameters. Algorithms that are strictly improving have a simple procedure, update the current point only if the candidate point is improving,

$$X_{k+1} = \begin{cases} V_{k+1} & \text{if } \ f(V_{k+1}) < f(X_k) \\ X_k & \text{otherwise.} \end{cases}$$

This type of improving algorithm may get trapped in a local optimum if the neighborhood, or procedure for generating candidate points is too restricted. One remedy is to sample a large neighborhood, as in VLSN or to draw from the entire feasible set. An example of this simple acceptance procedure is IHR, however its generator has a positive probability of sampling anywhere in the bounded feasible region, and so it converges in probability to the global optimum. Another possibility is to accept non-improving points, as in simulated annealing.

In simulated annealing [1, 32, 41], the update procedure is often called the Metropolis criterion, and the candidate point is accepted with a probability that reflects a Boltzmann distribution,

$$X_{k+1} = \begin{cases} V_{k+1} & \text{with probability } \ \min\left\{1, \exp\left(\frac{f(X_k)-f(V_{k+1})}{T_k}\right)\right\} \\ X_k & \text{otherwise.} \end{cases}$$

Notice that improving points are accepted with probability one, and the probability of accepting a worse point decreases as the temperature parameter cools, i.e. decreases to zero. The update of the temperature parameter, called the cooling schedule, has been studied in the literature [12, 23, 26, 36, 52, 60]. Typically the temperature parameter decreases monotonically, however recent research on dynamically heating and cooling the temperature in an Interacting Particle Algorithm [43] is discussed later in this article.

The update procedure for genetic algorithms and evolutionary programming is called the selection criterion. A straightforward scheme is to rank the population of points based on objective function value (also referred to as the fitness function), and select the top percentile of points to use in reproduction and recombination. However experience has shown that some diversity should be maintained to prevent premature convergence to a local optimum. The elitist strategy subdivides the population into three categories: the elite (best) solutions, the immigrant solutions (added to maintain diversity), and the crossover solutions [47]. Computational experience indicates that the population size impacts performance; if it is too small, the algorithm has difficulty finding the global optimum, and if it is too large, then it is inefficient and essentially pure random search. An effective population size for a particular problem is usually found through experimentation. Researchers have used Markov chain analysis to analyze the behavior, such as the expected waiting time, of genetic algorithms [14, 34].

Multi-start and Clustering Algorithms Algorithms may modify their generating and updating procedures per iteration. Consider multi-start; on one iteration a candidate point is generated according to a global sampling distribution on S, but the next iteration a local

search sampling distribution is used. Different combinations of algorithms can be created, such as generating uniformly distributed points in the global phase that initiate simulated annealing in the local phase, or use simulated annealing in the global search phase to initiate stochastic gradient search or nearest neighbor local searches.

An inefficiency in multi-start is that the same local optimum may be repeatedly discovered from many different starting points. This has motivated clustering methods where clusters are formed (often grown around a seed point) to predict whether a starting point is likely to lead to a new local optimum or one that has already been discovered. Then local searches are only initiated at promising candidate points [57, 67]. A related idea has led to linkage methods, which "link" points in the sample and essentially view clusters as trees, instead of spherical or ellipsoidal clusters. The well known linkage method called Multi Level Single Linkage [50, 51] and several variants including Random Linkage [37] are summarized in [57].

More recently, a multi-start heuristic for OptQuest/NLP, has been designed to operate on mixed integer nonlinear problems where the functions are differentiable with respect to the continuous variables [68]. They use scatter search (related to evolutionary programming) in the global search phase, and use filtering instead of clustering to decide when to initiate a local search. The nested partitioning method [61] is based on a partitioning of the feasible region that can be viewed as the global phase, coupled with local random search within the subregions.

Convergence in Probability Solis and Wets [63] provide a convergence proof, in probability, to the global minimum for general step size algorithms with conditions on the method of generating the step length and direction. Essentially, as long as the generator does not consistently ignore any region, then the algorithm will converge with probability one. Another convergence proof due to Bélisle [5] says that, even if the generator of an algorithm cannot reach any point in the domain in one iteration, if there is a means such as an acceptance probability to allow the algorithm to reach any point in a finite number of iterations, then the algorithm still converges with probability one to the global optimum. Stephens and Baritompa [65] go on to prove that an algorithm must either sample the entire region or use global information regarding the structure of the problem to provide convergence results. Examples of global information are bounds on the function, a Lipschitz constant, information on the level sets, number of local optima, functional form, and the global optimum itself. They conclude that using "global optimization heuristics is often far more practical than running general algorithms until the mathematically proven stopping criteria are satisfied" [65, page 587].

The generating and updating procedures for random search algorithms can be viewed as Markov chain Monte Carlo samplers, and then the algorithms can be interpreted as implicitly or explicitly sampling from a sequence of distributions. Markov chain theory has been applied to the analysis of several versions of these algorithms, including simulated annealing and genetic algorithms, to prove convergence. We next analyze the performance of three algorithms with explicit sampling distributions: pure random search, pure adaptive search, and annealing adaptive search, to gain insight into the distributions we could approximate with generating and updating procedures.

2 Performance of Pure Random Search, Pure Adaptive Search and Annealing Adaptive Search

The simplest and most obvious random search method is a "blind search" as discussed in [64], or called pure random search in [73]. *Pure random search* was first defined in 1958 by Brooks [9], and discussed later in [18, 19]. Pure random search (PRS) samples repeatedly from the feasible region S, typically according to a uniform sampling distribution. In the context of the generic random search algorithm, each candidate point is generated independently from a uniform distribution on S, and X_{k+1} is updated only if the candidate point is improving. It can be shown that pure random search converges to within an ϵ distance of the global optimum with probability one [63, 64, 73]. Pure random search is often the global phase in multi-start and clustering algorithms.

Even though pure random search converges in probability, it will take a long time. In order to describe the performance of PRS, we use $E[N(y_* + \epsilon)]$, the expected number of iterations until a point within ϵ distance of the global minimum is first sampled, as a measure of computational complexity. For PRS, an iteration generates one point uniformly on S and performs exactly one function evaluation, so this captures the majority of the computational effort. The random variable $N(y)$, the number of iterations until a point is first sampled with an objective function value of y or less, has a geometric distribution [17], where the probability of a "success" is $p(y)$, the probability of generating a point in the level set $S(y)$, where $S(y) = \{x \in S : f(x) \leq y\}$. Then, the expected number of iterations until a point is first sampled within ϵ distance of the optimum is

$$E[N(y_* + \epsilon)] = 1/p(y_* + \epsilon).$$

This relationship supports the natural intuition that, as the target region close to the global optimum gets small (i.e., $p(y_* + \epsilon)$ gets small), the expected number of iterations gets large (inversely proportional). The probability of failure to achieve $y_* + \epsilon$ after k iterations is $1 - p(y_* + \epsilon)^k$. The variance of the number of iterations until a point first lands in $S(y)$ is

$$\text{Var}(N(y)) = \frac{1 - p(y)}{p(y)^2}.$$

This supports numerical observations that pure random search (and other random search methods) experiences large variation; as the probability $p(y)$ decreases, the variance of $N(y)$ increases.

To gain insight into the performance of PRS, consider the TSP with N cities and subsequently $(N-1)!$ possible points in the domain. If there is a unique minimum, then $p(y_*) = 1/(N-1)!$ and the expected number of PRS iterations to first sample the minimum is $(N-1)!$, which explodes in N.

To illustrate the performance of PRS on a continuous domain problem, consider a global optimization problem where the domain S is an n-dimensional ball of radius 1, and the area within ϵ distance of the global optimum is a ball of radius ϵ, for $0 < \epsilon \leq 1$. Using a uniform sampling distribution on this problem, $p(y_* + \epsilon) = \epsilon^n$ for $0 < \epsilon \leq 1$ and the expected number of iterations until a sample point falls within the ϵ-ball is $(1/\epsilon)^n$, an exponential function of dimension.

To expand this example, it is shown in [73] that the expected number of PRS iterations on a global optimization problem where the objective function satisfies a Lipschitz condition with constant K is also exponential in the dimension of the problem,

$$E[N(y_* + \epsilon)] = (K/\epsilon)^n.$$

All random search algorithms are meant to improve upon PRS. However, the "No Free Lunch" theorems in Wolpert and Macready [70] show that, in essence, no single algorithm can improve on any other when averaged across all possible problems. As a consequence of these theorems, in order to improve upon PRS, we must either restrict the class of problems, have some prior information, or adapt the algorithm as properties of a specific problem are observed.

In contrast to pure random search, where each sample is independent and identically distributed, we next consider pure adaptive search, where each sample depends on the one immediately before it. Pure adaptive search (PAS) was introduced in [46] for convex programs and later analyzed for global optimization problems with Lipschitz continuous functions in [74] and for finite domains in [76]. It was generalized as hesitant adaptive search in [11, 72]. Pure adaptive search, by definition, generates a candidate point uniformly in the subset of the domain that is strictly improving, $V_{k+1} \in S(f(X_k)) = \{x : x \in S \text{ and } f(x) < f(X_k)\}$. It is an idealized algorithm to show potential performance for a random search algorithm. Whereas sampling from a uniform distribution in PRS over S is typically very easy (for example, if S is a hyperrectangle), sampling from a uniform distribution in PAS over $S(f(X_k))$ is very difficult in general. However, the analysis shows the value of being able to find points in the improving level set; the number of iterations of PAS is an exponential improvement over the number of iterations in PRS (Theorem 2.2, [73]). Additionally, for Lipschitz continuous objective functions, the expected number of iterations of pure adaptive search is *linear* in dimension (Theorem 2.9, [73]), with an analogous result for finite domains (Corollary 2.9, [73]). The linearity result for pure adaptive search implies that adapting the search to sample improving points is very powerful.

The idealistic linear property of PAS has inspired the design of several algorithms with the goal of approximating PAS, including Improving Hit-and-Run [75] which uses Hit-and-Run as a Markov chain Monte Carlo sampler embedded in an optimization context, and Grover Adaptive Search [10] which uses quantum computing for optimization.

We now describe one more idealistic algorithm, Annealing Adaptive Search, which is a way to abstract simulated annealing. Annealing adaptive search (AAS) differs from PAS in that the candidate points are generated from the entire feasible set S according to a Boltzmann distribution parameterized by temperature, whereas the sampling distribution in PAS only has positive support on nested level sets of S. The name *annealing adaptive search* (suggested in [71]) is used because the algorithm assumes points can be generated from exact Boltzmann distributions, which is a model of simulated annealing. The purpose of the AAS analysis is to gain understanding of simulated annealing and other stochastic algorithms.

The record values of AAS, first analyzed in [53], stochastically dominate those of PAS and hence inherit the ideal linear complexity of PAS. The number of sample points (including non-improving points) of AAS is also shown to be linear on average under certain conditions on the cooling schedule for the temperature parameter. The analysis motivated a cooling

schedule for simulated annealing which maintains a linear complexity in expected number of sample points on a class of (Lipschitz) objective functions over both continuous and discrete domains [60].

3 Cross-entropy and other Model-based Methods

In keeping with our generic random search algorithm, we now relate model-based methods to the generating and updating procedures. We view instance-based methods as emphasizing generating new points and subsequently updating them using algorithm parameters Θ_k, but now model-based methods change the emphasis to updating the algorithm parameters Θ_k and subsequently using them to generate new points. We contrast the two methods by comparing them to an idealized algorithm, such as PAS or AAS.

Consider PAS or AAS, where we want to sample points according to a sequence of probability density functions $h_k(x)$. In PAS the sampling distribution $h_k(x)$ is uniform on the level set associated with the best objective function value on the k^{th} iteration. In AAS the sampling distribution $h_k(x)$ is Boltzmann with a temperature parameter associated with the k^{th} iteration. Simulated annealing can be interpreted as a Markov chain sampling approach to approximate a series of AAS Boltzmann distributions, and IHR can be interpreted as a Markov chain sampling approach to approximate a series of PAS uniform distributions on level sets.

The approach of model-based methods is to approximate a sequence of target distributions $h_k(x)$ using the concept of importance sampling or sequential importance sampling [55, 56, 77]. In [77], model-based methods are shown to include ant colony optimization, stochastic gradient search, cross-entropy and estimation of distribution methods. The model reference adaptive search (MRAS) algorithm introduced in [29] also incorporates the idea of importance sampling and uses the information provided by the samples to update the parameters so the new samplings are biased toward the global optima.

Because sampling directly from $h_k(x)$ is difficult, the idea is to use a simple parametrized probability density function $g(x, \Theta)$. Choices for $g(x, \Theta)$ typically come from the natural exponential family, which include Gaussian, Poisson, binomial, geometric, and other multivariate forms [29, 55]. For example, when the Gaussian probability density function is used, then Θ consists of the mean vector and the covariance matrix. We now seek a parameter Θ_{k+1} that provides a good approximation to $h_k(x)$. This leads to the problem of minimizing the Kullback-Leibler divergence between $g(x, \Theta)$ and $h_k(x)$, that is

$$\Theta_{k+1} = \arg\inf_{\Theta} \int_S \ln\left(\frac{h_k(x)}{g(x,\Theta)}\right) h_k(x)dx.$$

The name of the cross-entropy method comes from this idea because the Kullback-Leibler divergence is also referred to as the *cross-entropy* between two probability density functions.

Now a new probability density function $h_{k+1}(x)$ is created by "tilting" $h_k(x)$ to bias it towards the improving regions in S using a performance function $G(x)$, as follows

$$h_{k+1}(x) = \frac{G_{y_{k+1}}(x)h_k(x)}{\int_S G_{y_{k+1}}(x)h_k(x)dx}.$$

The performance function $G(x)$ is assumed to be positive and relates to the objective function $f(x)$. One alternative for a performance function is $G_{y_{k+1}}(x) = \exp(f(x))\, I_{S(y_{k+1})}(x)$ where

$$I_{S(y_{k+1})}(x) = \begin{cases} 1 & \text{if } x \in S(y_{k+1}) \\ 0 & \text{otherwise} \end{cases}$$

is the indicator of the level set $S(y_{k+1})$. A sample of candidate points drawn from the sampling distribution $g(x, \Theta_{k+1})$ is used to determine a level set threshold y_{k+1}. In [29], y_{k+1} is estimated as a specified quantile of $f(x)$ with respect to $h_k(x)$.

Global convergence properties of these model-based methods appear in [29, 55, 77] with compelling numerical results.

4 Meta-control of Random Search Algorithms

To complete this article, we mention recent work on meta-control of random search algorithms. A meta-control methodology for prescribing parameters of a deterministic optimization algorithm was introduced in [35]. Extending the idea to random search algorithms, in [43] a meta-control methodology is derived to control the temperature parameter in the interacting particle algorithm.

The interacting particle algorithm in [42, 43] is based on Feynman-Kac systems in statistical physics [15, 16] and combines the ideas of simulated annealing with population-based algorithms. As such, it has a temperature parameter that influences the sampling distribution through acceptance/rejection. Instead of choosing a cooling schedule a priori, the meta-control methodology dynamically determines the temperature parameter by adapting it based on observed sample points. An optimal control problem that controls the evolution of the probability density function of the particles with the temperature parameter is defined to make the algorithm's search process satisfy specified performance criteria. The criteria in [43] include the expected objective function value of the particles, the spread of the particles, and the algorithm running time. Numerical results indicate improved performance by allowing heating and cooling of the temperature parameter through the meta-control methodology [43]. The meta-control methodology shares a common goal with cross-entropy and model-based methods; to provide a means to dynamically adapt algorithm parameters based on observed values in a closed-loop with feedback.

Random search algorithms provide a useful tool for practitioners to solve ill-structured global optimization problems. Combinations of proposed algorithms, with theoretical analyses, will lead to improved algorithms in the future.

References

[1] Aarts, E., and Korst, J. (1989), *Simulated Annealing and Boltzmann Machines - a Stochastic Approach to Combinatorial Optimization and Neural Computers*, Wiley, New York.

[2] Ahuja, R. K., Ergun, O., Orlin, J. B., and Punnen, A. P. (2002), "A Survey of Very Large-Scale Neighborhood Search Techniques," *Discrete Applied Mathematics*, 123, 75-102.

[3] Bäck, T., Fogel, D., and Michalewicz, Z. (1997), *Handbook of Evolutionary Computation*, IOP Publishing and Oxford University Press, New York, NY.

[4] Baumert, S., Ghate, A., Kiatsupaibul, S., Shen, Y., Smith, R. L., and Zabinsky, Z. B., Discrete Hit-and-Run for Generating Multivariate Distributions over Arbitrary Finite Subsets of a Lattice, forthcoming in *Operations Research*, 2009.

[5] Bélisle, C. J. P. (1992), "Convergence Theorems for a Class of Simulated Annealing Algorithms on $\mathbb{R}^d$," *J. Applied Probability*, 29, 885-895.

[6] Bélisle, C. J. P., Romeijn, H. E., and Smith, R. L. (1993), "Hit-and-Run Algorithms for Generating Multivariate Distributions," *Mathematics of Operations Research*, 18, 255-266.

[7] Blum, C., and Roli, A., (2003) "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," *ACM Computing Surveys*, 35(3), 268-308.

[8] Boender, C.G.E, and Romeijn, H.E. (1995), "Stochastic Methods," in *Handbook of Global Optimization*, edited by R. Horst and P. M. Pardalos, Kluwer Academic Publishers, Netherlands, 829-869.

[9] Brooks, S. H. (1958), "A Discussion of Random Methods for Seeking Maxima," *Operations Research*, 6, 244-251.

[10] Bulger, D., Baritompa, W. P., and Wood, G. R. (2003) "Implementing Pure Adaptive Search with Grover's Quantum Algorithm," *Journal of Optimization Theory and Applications*, 116, 517-529.

[11] Bulger, D. W., and Wood, G. R. (1998), "Hesitant Adaptive Search for Global Optimization," *Mathematical Programming*, 81, 89-102.

[12] Cohn, H., and Fielding, M. (1999), "Simulated annealing: searching for an optimal temperature schedule," *SIAM Journal on Optimization*, 9(3), 779-802.

[13] Davis, L. (1991), *Handbook of Genetic Algorithms*, Van Nostrand Reinhord, New York.

[14] De Jong, K. A., Spears, W. M., and Gordon, D.F. (1995), "Using Markov Chains to Analyze GAFOs," in *Foundations of Genetic Algorithms 3,* edited by D. Whitley, and M. Vose, Morgan Kaufmann, San Francisco, 115-137.

[15] Del Moral, P. (2004), *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*, Springer, New York.

[16] Del Moral, P., and Miclo, L. (1999), "On the convergence and applications of generalized simulated annealing," *SIAM Journal of Control and Optimization*, 37(4), 1222-1250.

[17] Devore, J. L. (1995), *Probability and Statistics for Engineering and the Sciences*, Fourth Edition, Wadsworth, Inc. Belmont, CA.

[18] Dixon, L. C. W., and Szegö, G. P. (1975), *Towards Global Optimization*, North-Holland, Amsterdam.

[19] Dixon, L. C. W., and Szegö, G. P. (1978), *Towards Global Optimization 2*, North-Holland, Amsterdam.

[20] Dorigo, M., and Stützle, T. (2004), *Ant colony optimization*, MIT Press, Cambridge, MA.

[21] Dyer, M. E., and Frieze, A. M. (1991), "Computing the Volume of Convex Bodies: A Case Where Randomness Provably Helps," *Proceedings of Symposia in Applied Mathematics*, 44, 123-169.

[22] Dyer, M., Frieze, A., and Kannan, R. (1991), "A random polynomial time algorithm for approximating the volume of convex bodies," *Journal of the ACM*, 38, 1-17.

[23] Fielding, M. (2000), "Simulated annealing with an optimal fixed temperature," *SIAM Journal on Optimization*, 11(2), 289-307.

[24] Glover, F., and Kochenberger, G. A. (2003), *Handbook of Metaheuristics*, International series in operations research & management science, 57, Kluwer Academic Publishers, Boston.

[25] Glover, F., and Laguna, M. (1993), "Tabu Search," in *Modern Heuristic Techniques for Combinatorial Problems,* edited by C. R. Reeves, Halsted Press, New York, 70-150.

[26] Hajek, B. (1988), "Cooling schedules for optimal annealing," *Math. Oper. Res.*, 13, 311-329.

[27] Horst, R., and Hoang, T. (1996), *Global Optimization: Deterministic Approaches*, Third Edition, Springer-Verlag, Berlin.

[28] Horst, R., and Pardalos, P. M. (1995), *Handbook of Global Optimization,* Kluwer Academic Publishers, Netherlands.

[29] Hu, J., Fu, M. C., and Marcus, S. I. (2007) "A Model Reference Adaptive Search Method for Global Optimization," *Operations Research*, 55, 549-568.

[30] Kennedy, J., and Eberhart, R. C. (1995), "Particle Swarm Optimization," *Proceedings of IEEE international conference on Neural Networks*, 4, 1942-1948.

[31] Kennedy, J., Eberhart, R. C. and Shi, Y. (2001), *Swarm Intelligence*, Morgan Kaufmann Publishers, San Fransisco, CA.

[32] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P. (1983), "Optimization by Simulated Annealing," *Science*, 20, 13 May, 671-680.

[33] Koehler, G. J. (1997), "New Directions in Genetic Algorithm Theory," *Annals of Operations Research*, 75, 49-68.

[34] Koehler, G. J. (1999), "Computing Simple GA Expected Waiting Times," *Proceedings of the Genetic and Evolutionary Computation Conference*, 795, 1999.

[35] Kohn, W., Zabinsky, Z. B., and Brayman, V. (2006), "Meta-Control of an optimization algorithm," *Journal of Global Optimization*, 34(2), 293-316.

[36] Locatelli, M. (2000), "Convergence of a simulated annealing algorithm for continuous global optimization," *Journal of Global Optimization*, 18, 219-234.

[37] Locatelli, M. and Schoen, F. (1999) "Random Linkage: a Family of Acceptance / Rejection Algorithms for Global Optimization," *Mathematical Programming*, 85(2), 379-396.

[38] Lovász, L. (1999), "Hit-and-Run Mixes Fast," *Mathematical Programming*, 86, 443-461.

[39] Luenberger, D. G. (1984), *Linear and Nonlinear Programming*, Second Edition, Addison-Wesley, Massachusetts.

[40] Mete, H. O., Shen, Y., Zabinsky, Z. B., Kiatsupaibul, S., and Smith, R. L., "Pattern Discrete and Mixed Hit-and-Run for Global Optimization," *Technical Report*, University of Washington, Seattle, WA, 2009.

[41] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953), "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, 21, 1087-1090.

[42] Molvalioglu, O., Zabinsky, Z. B., and Kohn, W. (2007), "Multi-particle simulated annealing," in *Models and Algorithms for Global Optimization*, edited by A. Törn, J. Zilinskas, and A. Zilinskas, 215-222, Springer, New York.

[43] Molvalioglu, O., Zabinsky, Z. B., and Kohn, W. (2009) "The interacting-particle algorithm with dynamic heating and cooling," *Journal of Global Optimization,* 43, 329-356.

[44] Mutseniyeks, V. A., and Rastrigin, L. (1964), "Extremal Control of Continuous Multiparameter Systems by the Method of Random Search," *Engineering Cybernetics*, 1, 82-90.

[45] Pardalos, P. M., and Romeijn, H. E. (2002), *Handbook of Global Optimization, Volume 2,* Kluwer Academic Publishers, Netherlands.

[46] Patel, N. R., Smith, R. L., and Zabinsky, Z. B. (1988), "Pure adaptive search in Monte Carlo optimization," *Mathematical Programming*, 4, 317-328.

[47] Rardin, R. L. (1998), *Optimization in Operations Research*, Prentice Hall, New Jersey.

[48] Rastrigin, L. A. (1960), "Extremal Control by the Method of Random Scanning," *Automation and Remote Control 21*, 891-896.

[49] Rastrigin, L. A. (1963), "The Convergence of the Random Method in the Extremal Control of a Many-parameter System," *Automation and Remote Control 24*, 1337-1342.

[50] Rinnooy Kan, A. H. G., and Timmer, G. T. (1987), "Stochastic Global Optimization Methods; part I: Clustering Methods," *Mathematical Programming,* 37, 27-56.

[51] Rinnooy Kan, A. H. G., and Timmer, G. T. (1987), "Stochastic Global Optimization Methods; part II: Multi Level Methods, *Mathematical Programming*, 37, 57-78.

[52] Romeijn, H. E., and Smith, R. L. (1994a), "Simulated Annealing for Constrained Global Optimization," *Journal of Global Optimization*, 5, 101-126.

[53] Romeijn, H. E., and Smith, R. L. (1994b), "Simulated Annealing and Adaptive Search in Global Optimization," *Probability in the Engineering and Informational Sciences*, 8, 571-590.

[54] Romeijn, H. E., Zabinsky, Z. B., Graesser, D. L., and Neogi, S. (1999), "New Reflection Generator for Simulated Annealing in Mixed-Integer/Continuous Global Optimization," *Journal of Optimization Theory and Applications*, 101(2), 403-427.

[55] Rubinstein, R. Y., and Kroese, D. P. (2004), *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*, Springer, New York.

[56] Rubinstein, R. Y., and Kroese, D. P. (2008), *Simulation and the Monte Carlo Method*, Second Edition, John Wiley and Sons, New Jersey.

[57] Schoen, F. (2002) "Two-phase Methods for Global Optimization," in *Handbook of Global Optimization, Volume 2*, edited by P. M. Pardalos, and H. E. Romeijn, Kluwer Academic Publishers, Netherlands, 151-177.

[58] Schrack, G., and Borowski, N. (1972), "An experimental comparison of three random searches," in *Numerical Methods For Nonlinear Optimization*, edited by F. Lootsma, Academic Press, London, 137-147.

[59] Schrack, G., and Choit, M. (1976), "Optimized Relative Step Size Random Searches," *Mathematical Programming 10*, 270-276.

[60] Shen, Y., Kiatsupaibul, S., Zabinsky, Z. B., and Smith, R. L. (2007), "An Analytically Derived Cooling Schedule for Simulated Annealing," *Journal of Global Optimization*, 38, 333-365.

[61] Shi, L., and Ólafsson, S. (2000), "Nested Partitions Method for Global Optimization," *Operations Research*, 48(3), 390-407.

[62] Smith, R. L. (1984), "Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions," *Operations Research*, 32, 1296-1308.

[63] Solis, F. J., and Wets, R. J.-B. (1981) "Minimization by Random Search Techniques," *Mathematics of Operations Research*, 6, 19-30.

[64] Spall, J. C. (2003), *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control*, Wiley, Hoboken, New Jersey.

[65] Stephens, C. P., and Baritompa, W. P. (1998), "Global Optimization Requires Global Information," *Journal of Optimization Theory and Applications,* 96(3), 575-588.

[66] Storn, R., and Price, K. (1997). "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, 11(4), 341-359.

[67] Törn, A., and Žilinskas, A. (1989), *Global Optimization*, Springer-Verlag, Germany.

[68] Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., and Marti, R. (2007), "Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization," *Informs Journal on Computing,* 19(3), 328-340.

[69] Vavasis, S. A. (1995), "Complexity Issues in Global Optimization: A Survey," in *Handbook of Global Optimization,* edited by R. Horst, and P. M. Pardalos, Kluwer Academic Publishers, Netherlands, 27-41.

[70] Wolpert, D. H., and Macready, W. G. (1997) "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, 1, 67-82.

[71] Wood, G. R., and Zabinsky, Z. B. (2002), "Stochastic Adaptive Search," in *Handbook of Global Optimization Volume 2,* edited by P. M. Pardalos and H. E. Romeijn, Kluwer Academic Publishers, Dordrecht, Netherlands, 231-249.

[72] Wood, G. R., Zabinsky, Z. B., and Kristinsdottir, B. P. (2001), "Hesitant adaptive search: the distribution of the number of iterations to convergence," *Mathematical Programming*, 89(3), 479-486.

[73] Zabinsky, Z. B. (2003), *Stochastic adaptive search for global optimization*, Kluwer Academic Publishers, Boston.

[74] Zabinsky, Z. B., and Smith, R. L. (1992), "Pure Adaptive Search in Global Optimization," *Mathematical Programming*, 53, 323-338.

[75] Zabinsky, Z. B., Smith, R. L., McDonald, J. F., Romeijn, H. E., and Kaufman, D. E. (1993), "Improving Hit and Run for Global Optimization," *Journal of Global Optimization*, 3, 171-192.

[76] Zabinsky, Z. B., Wood, G. R., Steel, M. A., and Baritompa, W. P. (1995), "Pure Adaptive Search for Finite Global Optimization," *Mathematical Programming*, 69, 443-448.

[77] Zlochin, M., Birattari, M., Meuleau, N., and Dorigo, M. (2004). "Model-Based Search for Combinatorial Optimization: A Critical Survey," *Annals of Operations Research*, 131, 373-395.