ASA OPTIONS

Lester Ingber

Abstract

Adaptive Simulated Annealing (ASA) is a C-language code that finds the best global fit of a nonlinear cost-function over a D-dimensional space. ASA has over 100 OPTIONS to provide robust tuning over many classes of nonlinear stochastic systems. These many OPTIONS help ensure that ASA can be used robustly across many classes of systems.

Keywords: simulated annealing, nonlinear systems

Most recent drafts are available as http://www.ingber.com/asa11_options.pdf

$Id: asa11_options,v 1.25 2011/06/15 22:00:47 ingber Exp ingber $

1. Introduction

Simulated annealing (SA) presents an optimization technique that can: (a) process cost functions possessing quite arbitrary degrees of nonlinearities, discontinuities, and stochasticity; (b) process quite arbitrary boundary conditions and constraints imposed on these cost functions; (c) be implemented quite easily with the degree of coding quite minimal relative to other nonlinear optimization algorithms; (d) statistically guarantee finding an optimal solution.

Adaptive Simulated Annealing (ASA) is a C-language code that finds the best global fit of a nonlinear cost-function over a D-dimensional space. The basic algorithm was originally published as Very Fast Simulated Reannealing (VFSR) in 1989 (Ingber, 1989), after two years of application on combat simulations. The code (Ingber, 1993a) can be used at no charge and downloaded from http://www.ingber.com/#ASA with mirrors at:
http://alumni.caltech.edu/~ingber
http://asa-caltech.sourceforge.net
https://code.google.com/p/adaptive-simulated-annealing .

ASA has over 100 OPTIONS to provide robust tuning over many classes of nonlinear stochastic systems. The current number as of this chapter is 152. These many OPTIONS help ensure that ASA can be used robustly across many classes of systems.

In the context of this book, it will be seen in the discussions that the "QUENCHing" OPTIONS are among the most important for controlling Adaptive Simulated Annealing. Fuzzy ASA algorithms in particular offer new ways of controlling how these QUENCHing OPTIONS may be applied across many classes of problems.

1.1. LICENSE and Contributions

The code originally was issued under a BSD-type License. This was changed to a form consistent with the less restrictive New BSD License
http://en.wikipedia.org/wiki/BSD_License
beginning with Version 28.1 in February 2011.

I have had several queries as to why I did not follow a GPL license. I felt and still feel, similar to many other people who make code available at no charge to others, that the GPL license is just too cumbersome and onerous. I have made my code available at no charge to anyone or any company, subject to very simple terms. If some user contributions do not quite fit into the code per se, I have put or referenced their contributions into the asa_contrib.txt or ASA-NOTES files. I do not think this has stymied people from contributing to the code.

For example, in http://www.ingber.com/asa_contrib.txt there are references to several major contributions made by other people, e.g., Matlab interface, RLAB interface, AMPL interface, and Haskell Interface, The ASA_PARALLEL OPTIONS were contributed as a team effort I led, as Principal Investigator of a 1994 National Science Foundation Parallelizing ASA and PATHINT Project (PAPP). The Editor of this book has contributed FUZZY_ASA OPTIONS (Oliveira, 2001; Oliveira, H.R. Petraglia & Petraglia, 2007; Oliveira, A. Petraglia & Petraglia, 2009). Another user referenced in http://www.ingber.com/asa_contrib.txt contributed explicit code used in ASA to help parallelize optimization of chip design.

The current list of CONTRIBUTORS in the ASA-CHANGES file that comes with code numbers 56. All these contributions have resulted in many versions of the code. The current list of VERSION DATES in the ASA-CHANGES file that comes with code numbers 586 since 1987. A few ASA papers showed how the code could be useful for many projects (Ingber, 1993b; Ingber, 1996a; Atiya *et al*, 2003).

1.2. Organization of Chapter

The next two sections give a short introduction to simulated annealing and to ASA. The first section discusses the theoretical foundations of ASA, and the second section discusses the practical implementation of ASA. The following section gives an overview and several approaches that consider why tuning is necessary in any sampling algorithm like SA, GA, etc. These issues have been addressed according to user feedback, i.e., what helps many users in many disciplines with a broad range of

experience to no experience. This work follows theoretical development of the algorithm that can be found in other ASA papers (Ingber, 1989; Ingber, 1993b; Ingber, 1996a).

Other sections that follow illustrate the use of OPTIONS are devoted Adaptive OPTIONS and Multiple Systems. The last section is the conclusion.

Most of this chapter has organized information that has collected on the use of the code since 1987, and is contained in some form in multiple files, e.g., ASA-README, ASA-NOTES, asa_contrib.txt, asa_examples.txt, etc.

2. Theoretical Foundations of Adaptive Simulated Annealing (ASA)

The unique aspect of simulated annealing (SA) is its property of (weak) ergodicity, permitting such code to statistically and reasonably sample a parameter space. Note that for very large systems, ergodicity is not an entirely rigorous concept when faced with the real task of its computation (Ma, 1985). In this chapter "ergodic" is used in a very weak sense, as it is not proposed, theoretically or practically, that all states of the system are actually to be visited.

2.1. Shades of simulated annealing

Even "standard" SA is not without its critics. Some negative features of SA are that it can: (A) be quite time-consuming to find an optimal fit, especially when using the "standard" Boltzmann technique; (B) be difficult to fine tune to specific problems, relative to some other fitting techniques; (C) suffer from "over-hype" and faddish misuse, leading to misinterpretation of results; (D) lose its ergodic property by misuse, e.g., by transforming SA into a method of "simulated quenching" (SQ) for which there is no statistical guarantee of finding an optimal solution. There also is a large and growing domain of SA-like techniques, which do not theoretically predict general statistical optimality, but which are extremely powerful for certain classes of problems.

There are many examples given in published papers addressing robust problems across many disciplines. There are many reviews of simulated annealing, comparisons among simulated annealing algorithms, and between simulated annealing and other algorithms (Johnson *et al*, 1987; Gelfand, 1987; van Laarhoven & Aarts, 1987; Collins *et al*, 1988; Ingber, 1993b; Ingber, 1996a).

It is important to compare the basic theoretic constraints of true SA with actual practice on a range of problems spanning many disciplines. This may help to address what may yet be expected in terms of better necessary conditions on SA to make it a more efficient algorithm, as many believe that the present sufficiency conditions are overly restrictive.

2.2. Critics of SA

The primary criticism is that it is too slow. This is partially addressed here by summarizing some work in appropriately adapting SQ to many problems. Another criticism is that it is "overkill" for many of the problems on which it is used. This is partially addressed here by pointing to much work demonstrating that it is not insignificant that many researchers are using SA/SQ because of the ease in which constraints and complex cost functions can easily be approached and coded.

There is another class of criticisms that the algorithm is too broadly based on physical intuition and is too short on mathematical rigor (Charnes & Wolfe, 1989). In some particular bitter and scathing critiques authors take offense at the lack of reference to other prior work (Pincus, 1970), the use of "metaphysical non-mathematical ideas of melting, cooling, and freezing" reference to the physical process of annealing as used to popularize SA (Kirkpatrick *et al*, 1983), and they give their own calculations to demonstrate that SA can be a very poor algorithm to search for global optima in some instances.

That there are undoubtedly other references that should be more regularly referenced is an objective issue that has much merit, with respect to SA as well as to other research projects. The other criticisms may be considered by some to be more subjective, but they are likely no more extreme than the use of SQ to solve for global optima under the protective umbrella of SA.

2.3. "Standard" simulated annealing (SA)

The Metropolis Monte Carlo integration algorithm (Metropolis *et al*, 1953) was generalized by the Kirkpatrick algorithm to include a temperature schedule for efficient searching (Kirkpatrick *et al*, 1983). A sufficiency proof was then shown to put an lower bound on that schedule as $1/\log(t)$, where t is an artificial time measure of the annealing schedule (Geman & Geman, 1984). However, independent credit usually goes to several other authors for independently developing the algorithm that is now recognized as simulated annealing (Pincus, 1970; Cerny, 1982).

2.4. Boltzmann annealing (BA)

Credit for the first simulated annealing is generally recognized as a Monte Carlo importance-sampling technique for doing large-dimensional path integrals arising in statistical physics problems (Metropolis *et al*, 1953). This method was generalized to fitting non-convex cost-functions arising in a variety of problems, e.g., finding the optimal wiring for a densely wired computer chip (Kirkpatrick *et al*, 1983). The choices of probability distributions described in this section are generally specified as Boltzmann annealing (BA) (Szu & Hartley, 1987).

The method of simulated annealing consists of three functional relationships.

1. $g(x)$: Probability density of state-space of D parameters $x = \{x^i; i = 1, D\}$.
2. $h(\Delta E)$: Probability for acceptance of new cost-function given the just previous value.
3. $T(k)$: schedule of "annealing" the "temperature" T in annealing-time steps k, i.e., of changing the volatility or fluctuations of one or both of the two previous probability densities.

The acceptance probability is based on the chances of obtaining a new state with "energy" E_{k+1} relative to a previous state with "energy" E_k,

$$\begin{aligned} h(\Delta E) &= \frac{\exp(-E_{k+1}/T)}{\exp(-E_{k+1}/T) + \exp(-E_k/T)} \\ &= \frac{1}{1 + \exp(\Delta E/T)} \\ &\approx \exp(-\Delta E/T) \ , \end{aligned} \tag{1}$$

where ΔE represents the "energy" difference between the present and previous values of the energies (considered here as cost functions) appropriate to the physical problem, i.e., $\Delta E = E_{k+1} - E_k$. This essentially is the Boltzmann distribution contributing to the statistical mechanical partition function of the system (Binder & Stauffer, 1985).

This can be described by considering: a set of states labeled by x, each with energy $e(x)$; a set of probability distributions $p(x)$; and the energy distribution per state $d\mathbf{(}e(x)\mathbf{)}$, giving an aggregate energy E,

$$\sum_x p(x) d\mathbf{(}e(x)\mathbf{)} = E \ . \tag{2}$$

The principle of maximizing the entropy, S,

$$S = -\sum_x p(x) \ln[p(x)/p(\bar{x})] \ , \tag{3}$$

where $\bar{x}$ represents a reference state, using Lagrange multipliers (Mathews & Walker, 1970) to constrain the energy to average value T, leads to the most likely Gibbs distribution $G(x)$,

$$G(x) = \frac{1}{Z} \exp\mathbf{(}-H(x)/T\mathbf{)} \ , \tag{4}$$

in terms of the normalizing partition function Z, and the Hamiltonian H operator as the "energy" function,

$$Z = \sum_x \exp\mathbf{(}-H(x)/T\mathbf{)} \ . \tag{5}$$

For such distributions of states and acceptance probabilities defined by functions such as $h(\Delta E)$, the equilibrium principle of detailed balance holds. I.e., the distributions of states before, $G(x_k)$, and after, $G(x_{k+1})$, applying the acceptance criteria, $h(\Delta E) = h(E_{k+1} - E_k)$ are the same:

$$G(x_k)h(\Delta E(x)) = G(x_{k+1}) \ . \tag{6}$$

This is sufficient to establish that all states of the system can be sampled, in theory. However, the annealing schedule interrupts equilibrium every time the temperature is changed, and so, at best, this must be done carefully and gradually.

An important aspect of the SA algorithm is to pick the ranges of the parameters to be searched. In practice, computation of continuous systems requires some discretization, so without loss of much generality for applications described here, the space will be assumed to be discretized. There are additional constraints that are required when dealing with generating and cost functions with integral values. Many practitioners use novel techniques to narrow the range as the search progresses. For example, based on functional forms derived for many physical systems belonging to the class of Gaussian-Markovian systems, one could choose an algorithm for g,

$$g(\Delta x) = (2\pi T)^{-D/2} \exp[-\Delta x^2/(2T)] \ , \tag{7}$$

where $\Delta x = x - x_0$ is the deviation of x from x_0 (usually taken to be the just-previously chosen point), proportional to a "momentum" variable, and where T is a measure of the fluctuations of the Boltzmann distribution g in the D-dimensional x-space. Given $g(\Delta x)$, it has been proven (Geman & Geman, 1984) that it suffices to obtain a global minimum of $E(x)$ if T is selected to be not faster than

$$T(k) = \frac{T_0}{\ln k} \ , \tag{8}$$

with T_0 "large enough."

A heuristic demonstration shows that this equation for T will suffice to give a global minimum of $E(x)$ (Szu & Hartley, 1987). In order to statistically assure, i.e., requiring many trials, that any point in x-space can be sampled infinitely often in annealing-time (IOT), it suffices to prove that the products of probabilities of not generating a state x IOT for all annealing-times successive to k_0 yield zero,

$$\prod_{k=k_0}^{\infty} (1 - g_k) = 0 \ . \tag{9}$$

This is equivalent to

$$\sum_{k=k_0}^{\infty} g_k = \infty \ . \tag{10}$$

The problem then reduces to finding $T(k)$ to satisfy this equation.

For BA, if $T(k)$ is selected to be the Boltzmann criteria above, then the generating distribution g above gives

$$\sum_{k=k_0}^{\infty} g_k \geq \sum_{k=k_0}^{\infty} \exp(-\ln k) = \sum_{k=k_0}^{\infty} 1/k = \infty \ . \tag{11}$$

Although there are sound physical principles underlying the choices of the Boltzmann criteria above (Metropolis *et al*, 1953), it was noted that this method of finding the global minimum in x-space was not limited to physics examples requiring *bona fide* "temperatures" and "energies." Rather, this methodology can be readily extended to any problem for which a reasonable probability density $h(\Delta x)$ can be formulated (Kirkpatrick *et al*, 1983).

2.5. Simulated quenching (SQ)

Many researchers have found it very attractive to take advantage of the ease of coding and implementing SA, utilizing its ability to handle quite complex cost functions and constraints. However, the long time of execution of standard Boltzmann-type SA has many times driven these projects to utilize a temperature schedule too fast to satisfy the sufficiency conditions required to establish a true (weak) ergodic search. A

logarithmic temperature schedule is consistent with the Boltzmann algorithm, e.g., the temperature schedule is taken to be

$$T_k = T_0 \frac{\ln k_0}{\ln k} , \tag{12}$$

where T is the "temperature," k is the "time" index of annealing, and k_0 is some starting index. This can be written for large k as

$$\Delta T = -T_0 \frac{\ln k_0 \Delta k}{k(\ln k)^2} , \; k \gg 1$$

$$T_{k+1} = T_k - T_0 \frac{\ln k_0}{k(\ln k)^2} . \tag{13}$$

However, some researchers using the Boltzmann algorithm use an exponential schedule, e.g.,

$$T_{k+1} = cT_k , \; 0 < c < 1$$

$$\frac{\Delta T}{T_k} = (c-1)\Delta k , \; k \gg 1$$

$$T_k = T_0 \exp((c-1)k) , \tag{14}$$

with expediency the only reason given. While perhaps someday some less stringent necessary conditions may be developed for the Boltzmann algorithm, this is not now the state of affairs. The question arises, what is the value of this clear misuse of the claim to use SA to help solve these problems/systems? Adaptive simulated annealing (ASA) (Ingber, 1989; Ingber, 1993a), in fact does justify an exponential annealing schedule, but only if a particular distribution is used for the generating function.

In many cases it is clear that the researchers already know quite a bit about their system, and the convenience of the SA algorithm, together with the need for some global search over local optima, makes a strong practical case for the use of SQ. In some of these cases, the researchers have been more diligent with regard to their numerical SQ work, and have compared the efficiency of SQ to some other methods they have tried. Of course, the point must be made that while SA's true strength lies in its ability to statistically deliver a true global optimum, there are no theoretical reasons for assuming it will be more efficient than any other algorithm that also can find this global optimum.

2.6. Fast annealing (FA)

Although there are many variants and improvements made on the "standard" Boltzmann algorithm described above, many textbooks finish just about at this point without going into more detail about other algorithms that depart from this explicit algorithm (van Laarhoven & Aarts, 1987). Specifically, it was noted that the Cauchy distribution has some definite advantages over the Boltzmann form (Szu & Hartley, 1987). The Cauchy distribution,

$$g(\Delta x) = \frac{T}{(\Delta x^2 + T^2)^{(D+1)/2}} , \tag{15}$$

has a "fatter" tail than the Gaussian form of the Boltzmann distribution, permitting easier access to test local minima in the search for the desired global minimum.

It is instructive to note the similar corresponding heuristic demonstration, that the Cauchy $g(\Delta x)$ statistically finds a global minimum. If the Boltzmann T is replaced by

$$T(k) = \frac{T_0}{k} , \tag{16}$$

then here

$$\sum_{k_0}^{\infty} g_k \approx \frac{T_0}{\Delta x^{D+1}} \sum_{k_0}^{\infty} \frac{1}{k} = \infty . \tag{17}$$

Note that the "normalization" of g has introduced the annealing-time index k, giving some insights into how to construct other annealing distributions. The method of FA is thus seen to have an annealing schedule exponentially faster than the method of BA. This method has been tested in a variety of problems (Szu & Hartley, 1987).

2.7. Adaptive simulated annealing (ASA)

In a variety of physical problems we have a D-dimensional parameter-space. Different parameters have different finite ranges, fixed by physical considerations, and different annealing-time-dependent sensitivities, measured by the derivatives of the cost-function at local minima. BA and FA have distributions that sample infinite ranges, and there is no provision for considering differences in each parameter-dimension; e.g., different sensitivities might require different annealing schedules. This prompted the development of a new probability distribution to accommodate these desired features (Ingber, 1989), leading to a variant of SA that in fact justifies an exponential temperature annealing schedule. These are among several considerations that gave rise to Adaptive Simulated Annealing (ASA). Full details are available by obtaining the publicly available source code (Ingber, 1993a).

ASA considers a parameter α_k^i in dimension i generated at annealing-time k with the range

$$\alpha_k^i \in [A_i, B_i] , \tag{18}$$

calculated with the random variable y^i,

$$\alpha_{k+1}^i = \alpha_k^i + y^i (B_i - A_i) ,$$

$$y^i \in [-1, 1] . \tag{19}$$

Define the generating function

$$g_T(y) = \prod_{i=1}^{D} \frac{1}{2(|y^i| + T_i)\ln(1 + 1/T_i)} \equiv \prod_{i=1}^{D} g_T^i(y^i) . \tag{20}$$

Its cumulative probability distribution is

$$G_T(y) = \int_{-1}^{y^1} \cdots \int_{-1}^{y^D} dy'^1 \cdots dy'^D \, g_T(y') \equiv \prod_{i=1}^{D} G_T^i(y^i) ,$$

$$G_T^i(y^i) = \frac{1}{2} + \frac{\mathrm{sgn}\,(y^i)}{2} \frac{\ln(1 + |y^i|/T_i)}{\ln(1 + 1/T_i)} . \tag{21}$$

y^i is generated from a u^i from the uniform distribution

$$u^i \in U[0, 1] ,$$

$$y^i = \mathrm{sgn}\,(u^i - \frac{1}{2}) T_i [(1 + 1/T_i)^{|2u^i - 1|} - 1] . \tag{22}$$

It is straightforward to calculate that for an annealing schedule for T_i

$$T_i(k) = T_{0i} \exp(-c_i k^{1/D}) , \tag{23}$$

a global minima statistically can be obtained. I.e.,

$$\sum_{k_0}^{\infty} g_k \approx \sum_{k_0}^{\infty} [\prod_{i=1}^{D} \frac{1}{2|y^i|c_i}] \frac{1}{k} = \infty . \tag{24}$$

It seems sensible to choose control over c_i, such that

$$T_{fi} = T_{0i} \exp(-m_i) \quad \text{when} \quad k_f = \exp n_i ,$$

$$c_i = m_i \exp(-n_i/D) , \tag{25}$$

where m_i and n_i can be considered "free" parameters to help tune ASA for specific problems.

It has proven fruitful to use the same type of annealing schedule for the acceptance function h as used for the generating function g, but with the number of acceptance points, instead of the number of generated points, used to determine the k for the acceptance temperature.

New parameters α^i_{k+1} are generated from old parameters α^i_k from

$$\alpha^i_{k+1} = \alpha^i_k + y^i(B_i - A_i) \ , \tag{26}$$

constrained by

$$\alpha^i_{k+1} \in [A_i, B_i] \ . \tag{27}$$

I.e., y^i's are generated until a set of D are obtained satisfying these constraints.

2.7.1. Reannealing

Whenever doing a multi-dimensional search in the course of a real-world nonlinear physical problem, inevitably one must deal with different changing sensitivities of the α^i in the search. At any given annealing-time, it seems sensible to attempt to "stretch out" the range over which the relatively insensitive parameters are being searched, relative to the ranges of the more sensitive parameters.

This can be by periodically rescaling the annealing-time k, essentially reannealing, e.g., every hundred or so acceptance-events, in terms of the sensitivities s_i calculated at the most current minimum value of the cost function, $\underline{L}$,

$$s_i = \partial \underline{L} / \partial \alpha^i \ . \tag{28}$$

In terms of the largest $s_i = s_{\max}$, ASA can reanneal by using a rescaling for each k_i of each parameter dimension,

$$k_i \rightarrow k'_i \ ,$$

$$T'_{ik'} = T_{ik}(s_{\max}/s_i) \ ,$$

$$k'_i = (\ln(T_{i0}/T_{ik'})/c_i)^D \ . \tag{29}$$

T_{i0} is set to unity to begin the search, which is ample to span each parameter dimension.

The acceptance temperature is similarly rescaled. In addition, since the initial acceptance temperature is set equal to a trial value of $\underline{L}$, this is typically very large relative to the global minimum. Therefore, when this rescaling is performed, the initial acceptance temperature is reset to the most current minimum of $\underline{L}$, and the annealing-time associated with this temperature is set to give a new temperature equal to the lowest value of the cost-function encountered to annealing-date.

Also generated are the "standard deviations" of the theoretical forms, calculated as $[\partial^2 \underline{L}/(\partial \alpha^i)^2]^{-1/2}$, for each parameter α_i. This gives an estimate of the "noise" that accompanies fits to stochastic data or functions. At the end of the run, the off-diagonal elements of the "covariance matrix" are calculated for all parameters. This inverse curvature of the theoretical cost function can provide a quantitative assessment of the relative sensitivity of parameters to statistical errors in fits to stochastic systems.

A few other twists can be added, and such searches undoubtedly will never be strictly by rote. Physical systems are so different, some experience with each one is required to develop a truly efficient algorithm.

2.7.2. Self optimization

Another feature of ASA is its ability to recursively self optimize its own Program Options, e.g., the c_i parameters described above, for a given system. An application is described below.

2.7.3. Quenching

Another adaptive feature of ASA is its ability to perform quenching. This is applied by noting that the temperature schedule above can be redefined as

$$T_i(k_i) = T_{0i} \exp(-c_i k_i^{Q_i/D}) ,$$

$$c_i = m_i \exp(-n_i Q_i/D) , \tag{30}$$

in terms of the "quenching factor" Q_i. The above proof fails if $Q_i > 1$ as

$$\sum_k \prod^D 1/k^{Q_i/D} = \sum_k 1/k^{Q_i} < \infty . \tag{31}$$

This simple calculation shows how the "curse of dimensionality" arises, and also gives a possible way of living with this disease. In ASA, the influence of large dimensions becomes clearly focused on the exponential of the power of k being $1/D$, as the annealing required to properly sample the space becomes prohibitively slow. So, if we cannot commit resources to properly sample the space ergodically, then for some systems perhaps the next best procedure would be to turn on quenching, whereby Q_i can become on the order of the size of number of dimensions.

The scale of the power of $1/D$ temperature schedule used for the acceptance function can be altered in a similar fashion. However, this does not affect the annealing proof of ASA, and so this may be used without damaging the (weak) ergodicity property.

2.8. VFSR and ASA

The above defines this method of adaptive simulated annealing (ASA), previously called very fast simulated reannealing (VFSR) (Ingber, 1989) only named such to contrast it the previous method of fast annealing (FA) (Szu & Hartley, 1987). The annealing schedules for the temperatures T_i decrease exponentially in annealing-time k, i.e., $T_i = T_{i0} \exp(-c_i k^{1/D})$. Of course, the fatter the tail of the generating function, the smaller the ratio of acceptance to generated points in the fit. However, in practice, when properly tuned, it is found that for a given generating function, this ratio is approximately constant as the fit finds a global minimum. Therefore, for a large parameter space, the efficiency of the fit is determined by the annealing schedule of the generating function.

A major difference between ASA and BA algorithms is that the ergodic sampling takes place in an $n + 1$ dimensional space, i.e., in terms of n parameters and the cost function. In ASA the exponential annealing schedules permit resources to be spent adaptively on reannealing and on pacing the convergence in all dimensions, ensuring ample global searching in the first phases of search and ample quick convergence in the final phases. The acceptance function $h(\Delta x)$ chosen is the usual Boltzmann form satisfying detailed balance, and the acceptance-temperature reannealing paces the convergence of the cost function to permit ergodic searching in the n-parameter space considered as the independent variables of the dependent cost function.

3. Practical Implementation of ASA

Details of the ASA algorithm are best obtained from the code itself and from published papers. There are three parts to its basic structure.

3.1. Generating Probability Density Function

In a D-dimensional parameter space with parameters p^i having ranges $[A_i, B_i]$, about the k'th last saved point (e.g., a local optima), p_k^i, a new point is generated using a distribution defined by the product of distributions for each parameter, $g^i(y^i; T_i)$ in terms of random variables $y^i \in [-1, 1]$, where $p_{k+1}^i = p_k^i + y^i(B_i - A_i)$, and "temperatures" T_i,

$$g^i(y^i; T_i) = \frac{1}{2(|y^i| + T_i) \ln(1 + 1/T_i)} . \tag{32}$$

The OPTIONS USER_GENERATING_FUNCTION permits using an alternative to this ASA distribution function.

3.2. Acceptance Probability Density Function

The cost functions, $C(p_{k+1}) - C(p_k)$, are compared using a uniform random generator, $U \in [0, 1)$, in a "Boltzmann" test: If

$$\exp[-(C(p_{k+1}) - C(p_k))/T_{\rm cost}] > U \ , \tag{33}$$

where $T_{\rm cost}$ is the "temperature" used for this test, then the new point is accepted as the new saved point for the next iteration. Otherwise, the last saved point is retained. The OPTIONS USER_ACCEPT_ASYMP_EXP or USER_ACCEPT_THRESHOLD permit using alternatives to this Boltzmann distribution function.

3.3. Reannealing Temperature Schedule

The annealing schedule for each parameter temperature, T_i from a starting temperature T_{i0}, is

$$T_i(k_i) = T_{0i} \exp(-c_i k_i^{1/D}) \ . \tag{34}$$

The annealing schedule for the cost temperature is developed similarly to the parameter temperatures. However, the index for reannealing the cost function, $k_{\rm cost}$, is determined by the number of accepted points, instead of the number of generated points as used for the parameters. This choice was made because the Boltzmann acceptance criteria uses an exponential distribution that is not as fat-tailed as the ASA distribution used for the parameters. This schedule can be modified using several OPTIONS. In particular, the Pre-Compile OPTIONS USER_COST_SCHEDULE permits quite arbitrary functional modifications for this annealing schedule, and the Pre-Compile OPTIONS

As determined by the Program Options selected, the parameter "temperatures" may be periodically adaptively reannealed, or increased relative to their previous values, using their relative first derivatives with respect to the cost function, to guide the search "fairly" among the parameters.

As determined by the Program Options selected, the reannealing of the cost temperature resets the scale of the annealing of the cost acceptance criteria as

$$T_{\rm cost}(k_{\rm cost}) = T_{0\ {\rm cost}} \exp(-c_{\rm cost} k_{\rm cost}^{1/D}) \ . \tag{35}$$

The new $T_{0\ {\rm cost}}$ is taken to be the minimum of the current initial cost temperature and the maximum of the absolute values of the best and last cost functions and their difference. The new $k_{\rm cost}$ is calculated taking $T_{\rm cost}$ as the maximum of the current value and the absolute value of the difference between the last and best saved minima of the cost function, constrained not to exceed the current initial cost temperature. This procedure essentially resets the scale of the annealing of the cost temperature within the scale of the current best or last saved minimum.

This default algorithm for reannealing the cost temperature, taking advantage of the ASA importance sampling that relates most specifically to the parameter temperatures, while often is quite efficient for some systems, may lead to problems in dwelling too long in local minima for other systems. In such case, the user may also experiment with alternative algorithms effected using the Reanneal_Cost OPTIONS. For example, ASA provides an alternative calculation for the cost temperature, when Reanneal_Cost < -1 or > 1, that periodically calculates the initial and current cost temperatures or just the initial cost temperature, resp., as a deviation over a sample of cost functions.

These reannealing algorithms can be changed adaptively by the user, e.g., by using USER_REANNEAL_COST and USER_REANNEAL_PARAMETERS.

3.4. QUENCH_PARAMETERS=FALSE

This OPTIONS permits you to alter the basic algorithm to perform selective "quenching," i.e., faster temperature cooling than permitted by the ASA algorithm. This can be very useful, e.g., to quench the system down to some region of interest, and then to perform proper annealing for the rest of the run. However, note that once you decide to quench rather than to truly anneal, there no longer is any statistical guarantee of finding a global optimum.

Once you decide you can quench, there are many more alternative algorithms you might wish to choose for your system, e.g., creating a hybrid global-local adaptive quenching search algorithm, e.g., using

USER_REANNEAL_PARAMETERS. Note that just using the quenching OPTIONS provided with ASA can be quite powerful, as demonstrated in the http://www.ingber.com/asa_examples.txt file.

Setting QUENCH_PARAMETERS to TRUE can be extremely useful in very large parameter dimensions; see the ASA-NOTES file under the section on Quenching.

Many parameters can be conveniently read in from the asa_opt file. E.g., User_Quench_Cost_Scale and User_Quench_Param_Scale all are read in if OPTIONS_FILE_DATA, QUENCH_COST, and QUENCH_PARAMETERS are TRUE.

3.5. QUENCH_COST=FALSE

If QUENCH_COST is set to TRUE, the scale of the power of $1/D$ temperature schedule used for the acceptance function can be altered in a similar fashion to that described above when QUENCH_PARAMETERS is set to TRUE. However, note that this OPTIONS does not affect the annealing proof of ASA, and so this may be used without damaging the statistical ergodicity of the algorithm. Even greater functional changes can be made using the Pre-Compile OPTIONS:
USER_COST_SCHEDULE
USER_ACCEPT_ASYMP_EXP
USER_ACCEPT_THRESHOLD
USER_ACCEPTANCE_TEST

If QUENCH_COST=TRUE User_Quench_Cost_Scale must be defined.

This can have the effect of User_Quench_Param_Scale appear contrary, as the effects on the temperatures from the temperature scales and the temperature indexes can have opposing effects. However, these defaults are perhaps most intuitive when the User_Quench_Param_Scale are on the order of the parameter dimension.

When
QUENCH_PARAMETERS=TRUE
QUENCH_PARAMETERS_SCALE=FALSE
only the temperature indexes are affected by User_Quench_Param_Scale. The same effect could be managed by raising Temperature_Anneal_Scale to the appropriate power, but this may not be as convenient.

3.6. QUENCH_COST_SCALE=TRUE

When QUENCH_COST is TRUE, if QUENCH_COST_SCALE is TRUE, then the temperature scale and the temperature index are affected by User_Quench_Cost_Scale. This can have the effect of User_Quench_Cost_Scale appear contrary, as the effects on the temperature from the temperature scale and the temperature index can have opposing effects. However, these defaults are perhaps most intuitive when User_Quench_Cost_Scale is on the order of the parameter dimension.

When QUENCH_COST is TRUE, if QUENCH_COST_SCALE is FALSE, only the temperature index is affected by User_Quench_Cost_Scale. The same effect could be managed by raising Temperature_Anneal_Scale to the appropriate power, but this may not be as convenient.

4. Tuning Guidelines

4.1. The Necessity for Tuning

I am often asked how I can help someone tune their system, and they send me their cost function or a list of the ASA OPTIONS they are using. Most often, the best help I can provide is based on my own experience that nonlinear systems typically are non-typical. In practice, that means that trying to figure out the nature of the cost function under sampling in order to tune ASA (or likely to similarly tune a hard problem under any sampling algorithm), by examining just the cost function, likely will not be as productive as generating more intermediate printout, e.g., setting ASA_PRINT_MORE to TRUE, and looking at this output as a “grey box” of insight into your optimization problem. Larger files with more information is provided by setting ASA_PIPE_FILE to TRUE. Treat the output of ASA as a simulation

in the ASA parameter space, which usually is quite a different space than the variable space of your system.

For example, you should be able to see where and how your solution might be getting stuck in a local minima for a very long time, or where the last saved state is still fluctuating across a wide portion of your state space. These observations should suggest how you might try speeding up or slowing down annealing/quenching of the parameter space and/or tightening or loosening the acceptance criteria at different stages by modifying the OPTIONS, e.g., starting with the OPTIONS that can be easily adjusted using the asa_opt file.

The ASA-NOTES file that comes with the ASA code provides some guidelines for tuning that may provide some insights, especially the section Some Tuning Guidelines. An especially important guide is to examine the output of ASA at several stages of sampling, to see if changes in parameter and temperatures are reasonably correlated to changes in the cost function. Examples of useful OPTIONS and code that often give quick changes in tuning in some problems are in the file http://www.ingber.com/asa_examples.txt under WWW. Some of the reprint files of published papers in the ingber.com provide other examples in harder systems, and perhaps you might find some examples of harder systems using ASA similar to your own in http://www.ingber.com/asa_papers.html under WWW. This is the best way to add some Art to the Science of annealing.

While the upside of using ASA is that it has many OPTIONS available for tuning, derived in large part from feedback from many users over many years, making it extremely robust across many systems, the downside is that the learning curve can be steep especially if the default settings or simple tweaking in asa_opt do not work very well for your particular system, and you then must turn to using more ASA OPTIONS. Most of these OPTIONS have useful guides in the ASA_TEMPLATEs in asa_usr.c, as well as being documented here. If you really get stuck, you may consider working with someone else who already has climbed this learning curve and whose experience might offer quick help.

Tuning is an essential aspect of any sampling algorithm if it is to be applied to many classes of systems. It just doesn't make sense to compare sampling algorithms unless you are prepared to properly tune each algorithm to each system being optimized or sampled.

4.2. Construction of the Code

I sometimes get a query like:

> "I used your ASA code some years ago with good results and want to thank you for providing it.
>
> However even back then i noticed that it was in urgent need of a good refactoration, as described in http://en.wikipedia.org/wiki/Refactor .
>
> I encourage you to go over your code and split it up in more readable chunks. today's compilers are pretty good at optimizing the result so it will not impact your programs performance.
>
> Again, thank you very much for your excellent program."

My reply is typically along these lines:

> "When I first wrote the code it was in broken into multiple files which were easy to take care of. I made the decision, which feedback has shown to be a good one, to make the code look less formidable to many users by aggregating the code into just a few files. The code is used widely across many disciplines, but often by expert people or groups without computer science skills, and often tuning can be accomplished by tweaking the parameter file and not having to deal with the .c files very much.
>
> Even if I choose to keep just a few files, I just do not have the time to rewrite the code into better code similar to how I write code now, 20 years later (I first wrote the VFSR code in 1987). However, for me at least, the structure of the code makes it very easy to maintain, and I been able to be responsive to any major changes that might come up. The ASA-CHANGES files reflects this.

> I have led teams of extremely bright and competent math-physics and computer-science people in several disciplines over the years, and I have also seen how code that may be written in exemplary languages, whether C, Java, C++, python, etc., nonetheless can be rotten to maintain if it is not written in a "functional" manner that better reflects the underlying algebra or physical process, e.g., as most people would program in an algebraic language like Macsyma/Maxima, Maple, etc. In many of these projects, we had no problem using ASA. This does not excuse a lot of the clumsy writing in ASA, but it does reflect on the difference between code that is just well written but not flexible and robust to maintain.
>
> By now, ASA represents a lot of feedback from thousands of users. A major strength of the code is that it has well over 100 tuning OPTIONS, albeit in many case only a few are usually required. This is the nature of sampling algorithms, and I have broken out all such code-specific parameters into a top-level meta-language that is easy for an end-user to handle. Other very good sampling algorithms do not give such robust tuning, and too often do not work on some complex systems for some users just for this reason. This also has added a lot of weight to the code, but since most of these ASA OPTIONS are chosen at pre-compile time, this does not affect the executables in typical use. I have had at least half a dozen exceptional coders start to rewrite the code into another language, e.g., C++, Java, Matlab, etc., but they gave up when faced with integrating all the ASA OPTIONS. (There is no way I could influence them to start or stop such projects.) I think all these OPTIONS are indeed necessary for such a generic code.
>
> I very much appreciate your writing to me."

The OPTIONS are not just a way of compiling in only code that may be needed for systems so it can run efficiently. The OPTIONS provide a clear meta-language for users to understand how to adjust and tune the code for their own needs. Indeed, there are several OPTIONS that provide hooks for users to insert their own generating and acceptance distribution functions. This leads to a transparency of the code to end-users, at the expense of muddling the code for object-oriented coders.

4.3. Motivations for Tuning Methodology

Nonlinear systems are typically not typical, and so it is difficult if not impossible to give guidelines for ASA defaults similar to what you might expect for "canned" quasi-linear systems. I have tried to prepare the ASA-README to give some guidelines, and if all else fails you could experiment a bit using a logical approach with the SELF_OPTIMIZE OPTIONS. I still advise some experimentation that might yield a bit of insight about a particular system. In many case, the best approach is probably a "blend": Make a guess or two, then fine-tune the guesses with SELF_OPTIMIZE in some rather finer range of the parameter(s). The reason this is slow is because ASA does what you expect it to do: It truly samples the space. When SELF_OPTIMIZE is turned on, for each call of the top-level ASA parameters selected, the "inner" shell of your system's parameters are optimized, and this is performed for an optimization of the "outer" top-level shell of ASA parameters. If you find that indeed this is a necessary and valuable approach to your problem, then one possible short cut might be to turn on Quenching for the outer shell.

The ASA proof of statistical convergence to a global optimal point gives sufficient, not necessary, conditions. This still is a pretty strong statement since one can only importance-sample a large space in a finite time. Note that some spaces would easily require CPU times much greater than the lifetime of the universe to sample all points. If you "tucked away" a "pathological" singular optimal point in an otherwise "smooth" space, indeed ASA might have to run "forever." If the problem isn't quite so pathological, you might have to slow down the annealing, to permit ASA to spend more time at each scale to investigate the finer scales; then, you would have to explore some other OPTIONS. This could be required if your problem looks different at different scales, for then you can often get trapped in local optima, and thus ASA could fail just as any other "greedy" quasi-Newton algorithm.

Because of its exponential annealing schedule, ASA does converge at the end stages of runs quite well, so if you start with your setup akin to this stage, you will search for a very long time (possibly beyond your machine's precision to generate temperatures) to get out. Or, if you start with too broad a search, you will spin your wheels at first before settling down to explore multiple local optima.

ASA has demonstrated many times that it is more efficient and gets the global point better than other importance-sampling techniques, but this still can require "tuning" some ASA OPTIONS. E.g., as mentioned in the ASA-README, a quasi-Newton algorithm should be much more efficient than ASA for a parabolic system.

4.4. Some Rough But Useful Guidelines

Here are some crude guidelines that typically have been useful to tune many systems. At least ASA has a formal proof of convergence to the global minimum of your system. However, no sampling proof is general enough for all systems to guarantee this will take place within your lifetime. This is where the true power of ASA comes into play as the code provides many tuning OPTIONS, most which can be applied adaptively at any time in the run, to give you tools to tune your system to provide reasonably efficient optimizations. Depending on your system, this may be easy or hard, possibly taxing anyone's intuitive and analytic capabilities.

In general, respect the optimization process as a simulation in parameter space. The behavior of a system in this space typically is quite different from the system defined by other variables in the system.

(a) Three Stages of Optimization It is useful to think of the optimization process as having three main stages: initial, middle and end. In the initial stage you want to be sure that ASA is jumping around a lot, visiting all regions of the parameter space within the bounds you have set. In the end stage you want to be sure that the cost function is in the region of the global minimum, and that the cost function as well as the parameter values are being honed to as many significant figures as required. The middle stage typically can require the most tuning, to be sure it smoothly takes the optimization from the initial to the end stage, permitting plenty of excursions to regularly sample alternative regions/scales of the parameter space.

(b) Tuning Information Keep ASA_PRINT_MORE set to TRUE during the tuning process to gather information in asa_out whenever a new accepted state is encountered.

If you have ASA_PIPE and/or ASA_PIPE_FILE set to TRUE, additional information (in relatively larger files) is gathered especially for purposes of graphing key information during the run. Graphical aids can be indispensable for gaining some intuition about your system.

If ASA_SAVE_OPT is set to TRUE then you have the ability to restart runs from intermediate accepted states, without having to reproduce a lot of the original run each time you wish to adaptively change some OPTIONS after a given number of accepted or generated states.

(c) Parameter Temperatures As discussed above in the section Parameter-Temperature Scales, the temperature schedule is determined by T_{0i} , c_i, k_i, Q_i, and D. The default is to have all these the same for each parameter temperature.

Note that the sensitivity of the default parameter distributions to the parameter temperatures is logarithmic. Therefore, middle stage temperatures of 10E-6 or 10E-7 still permit very large excursions from the last local minima to visit new generated states. Typically (of course depending on your system), values of 10E-10 are appropriate for the end stage of optimization.

It is advisable to start by changing the c_i to get a reasonable temperature schedule throughout the run. If it becomes difficult to do this across the 3 stages, work with the Q_i QUENCH_PARAMETERS as these provide different sensitivities at different stages. Generally, it is convenient to use the c_i to tune the middle stage, then add in Q_i modifications for the end stage. As long as the sum $Q_i \leq 1$, then the sampling proof is intact. However, once you are sure of the region of the global minima, it can be convenient to turn on actual quenching wherein sum $Q_i > 1$.

Turning on Reanneal_Parameters can be very useful for some systems to adaptively adjust the temperatures to different scales of the system.

(d) Cost Temperature Note that the sensitivity of the default cost distribution to the cost temperatures is exponential.

In general, you would like to see the cost temperatures throughout the run be on the scale of the difference between the best and last generated states, where the last generated state in the run is at the last local minima from which new states are explored. Therefore, pay careful attention to these values. Note that the last generated state is set to the most recently accepted state, and if the recently accepted state also is

the current best state then the last generated state will be so reported. Therefore, this sensitivity to the last generated state works best during parts of the run where the code is sampling alternate multiple minima.

The default is to baseline the cost temperature scale to the default parameter temperature scale, using Cost_Parameter_Scale_Ratio (default = 1). It is advisable to first tune your parameter temperature schedule using Temperature_Ratio_Scale, then to tune your cost temperature schedule using Cost_Parameter_Scale_Ratio. If it becomes difficult to do this across the 3 stages, work with the Q QUENCH_COST as this provides a different sensitivity at a different stage. Generally, it is convenient to use the c scale via Cost_Parameter_Scale_Ratio to tune the middle stage, then add in Q modifications for the end stage.

Turning on Reanneal_Cost can be very useful for some systems to adaptively adjust the temperature to different scales of the system.

(e) Large Parameter Dimensions As the number of parameter dimensions D increases, you may see that your temperatures are changing more than you would like with respect to D. The default is to keep the parameter exponents of the k_i summed to 1 with each exponent set to $1/D$.

The effective scale of the default exponential decay of the temperatures is proportional to $ck^{-Q/D}$, so smaller D gives smaller decay rates for the same values of c, k and Q. Modifications to this behavior of the parameter and cost temperatures are easily made by altering the Q_i and Q, resp., as Q_i, Q and D enter the code as Q_i/D and Q/D, resp.

The scales c are set as c = - log (Temperature_Ratio_Scale) exp (-log (Temperature_Anneal_Scale) (Q/D). Therefore, the sensitivity of c to D can be controlled by modifying Temperature_Anneal_Scale or Q.

4.5. Quenching

If you have a large parameter space, and if a “smart” quasi-local optimization code won’t work for you, then any true global optimization code will be faced with the “curse of dimensionality”. I.e., global optimization algorithms must sample the entire space, and even an efficient code like ASA must do this. As mentioned in the ASA-README, there are some features to explore that might work for your system.

SQ techniques like genetic algorithms (GA) obviously are important and are crucial to solving many systems in time periods much shorter than might be obtained by standard SA. In ASA, if annealing is forsaken, and Quenching turned on, voiding the proof of sampling, remarkable increases of speed can be obtained, apparently sometimes even greater than other “greedy” algorithms.

In large D space, this can be especially useful if the parameters are relatively independent of each other, by noting that the arguments of the exponential temperature schedules are proportional to $k^{Q/D}$. Then, you might do better thinking of changing Q/D in fractional moves, instead of only small deviations of Q from 1.

For example, in http://www.ingber.com/asa92_saga.pdf, along with 5 GA test problems from the UCSD GA archive, another harder problem (the ASA_TEST problem that comes with the ASA code) was used. As reported in http://www.ingber.com/asa93_sapvt.pdf, Quenching was applied to this harder problem. The resulting SQ code was shown to speed up the search by as much as a factor of 86 (without even attempting to see if this could be increased further with more extreme quenching). In the asa_examples.txt file, even more dramatic efficiencies were obtained. This is a simple change of one number in the code, turning it into a variant of SQ, and is not equivalent to tuning any of the other many ASA options, e.g., like SELF_OPTIMIZE, USER_COST_SCHEDULE, etc. Note that SQ will not suffice for all systems; several users of ASA reported that Quenching did not find the global optimal point that was otherwise be found using the correct SA algorithm.

As mentioned in the ASA-README, note that you also can use the Quenching OPTIONS quite differently, to slow down the annealing process by setting User_Quench_Param_Scale to values less than 1. This can be useful in problems where the global optimal point is at a quite different scale from other local optima, masking its presence. This likely might be most useful for low dimensional problems where the CPU time incurred by slower annealing might not be a major consideration.

Once you decide you can quench, there are many more alternative algorithms you might wish to choose for your system, e.g., creating a hybrid global-local adaptive quenching search algorithm, e.g., using USER_REANNEAL_PARAMETERS. Note that just using the quenching OPTIONS provided with ASA can be quite powerful, as demonstrated in the asa_examples.txt file.

4.6. Options for Large Spaces

For very large parameter-space dimensions, the following guide is useful if you desire to speed up the search:

Pre-Compile Options:
add USER_REANNEAL_PARAMETERS=TRUE
add USER_COST_SCHEDULE=TRUE
add ASA_PRINT_INTERMED=FALSE
SMALL_FLOAT may have to be decreased
set QUENCH_PARAMETERS to TRUE [negates SA sampling if $Q > 1$]
set QUENCH_COST to TRUE
Perhaps set QUENCH_PARAMETERS_SCALE and QUENCH_COST_SCALE to FALSE

Program Options:
set Curvature_0 to TRUE
decrease Temperature_Ratio_Scale
increase Cost_Parameter_Scale_Ratio
increase Maximum_Cost_Repeat
decrease Acceptance_Frequency_Modulus
decrease Generated_Frequency_Modulus

If the parameter space dimension, D, is huge, e.g., 256x256=65536, then the exponential of the generating or acceptance index to the $1/D$ power hardly changes over even a few million cycles. True annealing in such huge spaces can become prohibitively slow as the temperatures will hardly be diminished over these cycles. This “curse of dimensionality” will face any algorithm seeking to explore an unknown space. Then, the QUENCH_PARAMETERS and QUENCH_COST OPTIONS should be tried.

However, note that slowing down annealing sometimes can speed up the search by avoiding spending too much time in some local optimal regions.

4.7. Shunting to Local Codes

I have always maintained in e-mails and in VFSR/ASA publications since 1987, that SA techniques are best suited for approaching complex systems for which little or no information is available. When the range of a global optima is discovered, indeed it may be best to then turn to another algorithm. I have done this myself in several papers, shunting over to a quasi-local search, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, to “polish” off the last 2 or 3 decimals of precision, after I had determined just what final level of precision was acceptable. In the problems where I shunted to BFGS, I simply used something the value of Cost_Precision or Limit_Acceptances (which were pretty well correlated in some problems) to decide when to shunt over. (I got terrible results if I shunted over too quickly.) However, that was before the days I added OPTIONS like USER_COST_SCHEDULE and USER_ACCEPTANCE_TEST, and if and when I redo some of those calculations I will first experiment adaptively using these to account for different behaviors of my systems at different scales.

When FITLOC is set to TRUE, three modified simplex subroutines, not requiring derivatives of cost functions, become active to perform a local fit after leaving asa ().

4.8. Judging Importance-Sampling

If the cost function is plotted simply as a function of decreasing temperature(s), often the parameter space does appear to be continually sampled in such a plot, but the plot is misleading. That is, there really is importance sampling taking place, and the proof of this is to do a log-log plot of the cost function versus the number of generated states. Then you can see that if the temperature schedule is not enforced you

will have a poor search, if quenching is turned on you will get a faster search (though you may miss the global minimum), etc. You can test these effects using quenching and "reverse quenching" (slowing down the annealing); it likely would be helpful to set:
QUENCH_COST and QUENCH_PARAMETERS to TRUE
QUENCH_PARAMETERS_SCALE and QUENCH_COST_SCALE to FALSE
perhaps NO_PARAM_TEMP_TEST and NO_COST_TEMP_TEST to TRUE

The point is that the ASA distribution is very fat-tailed, and the effective widths of the parameters being searched change very slowly with decreasing parameter temperatures; the trade-off is that the parameter temperatures may decrease exponentially and still obey the sampling proof. Thus, the experience is that ASA finds global minimum when other sampling techniques fail, and it typically finds the global minimum faster than other sampling techniques as well.

Furthermore, the independence of cost and parameter temperatures permits additional tuning of ASA in many difficult problems. While the decreasing parameter temperatures change the way the parameter states are generated, the decreasing cost temperature changes the way the generated states are accepted. The sensitivity to the acceptance criteria to the cost temperature schedule can be very important in many systems. An examination of a few runs using ASA_PRINT_MORE set to TRUE can reveal premature holding onto local minimum or not enough holding time, etc., requiring tuning of some ASA OPTIONS.

4.9. User References

Collaborators and I have published some papers in several disciplines that have used or expanded the use of ASA (Ingber, 1990; Ingber & Sworder, 1991; Ingber, Fujio & Wehner, 1991; Ingber, 1991; Ingber, 1992; Ingber, 1993b; Ingber, 1993c; Ingber, 1996c; Ingber, 1996b; Ingber, 1996a; Ingber, 1997; Bowman & Ingber, 1997; Ingber, 1998a; Ingber, 1998b; Ingber, 2001a; Ingber & Mondescu, 2001; Ingber, 2001b; Ingber, 2001c; Ingber, 2001d; Ingber & Mondescu, 2003; Atiya *et al*, 2003; Ingber, 2005; Ingber, 2006; Ingber, 2007a; Ingber, 2007b; Ingber, 2008a; Ingber, 2008b; Ingber, 2009; Ingber, 2010a; Ingber, 2010b).

The file http://www.ingber.com/asa_papers.html contains a short list of users who have sent me their papers using ASA. Many other users also have had to list ASA as a tool since it was used in the patents. That file also gives URLs to search patent filings for the use of ASA. The results reveal its use in many disciplines and companies.

5. Adaptive OPTIONS

5.1. VFSR

The first VFSR code (Ingber, 1989) added adaptive options by reannealing, i.e., increasing rather than decreasing, the temperature schedules for parameters and the cost function, to enable easier passage through multi-dimensional spaces en route to finding global optima. Of several such OPTIONS, most effective on many systems are Temperature_Ratio_Scale, Cost_Parameter_Scale_Ratio, and Temperature_Anneal_Scale.

5.2. ASA_FUZZY

The Editor of this book contributed ASA_FUZZY code to ASA, to help guide QUENCHing OPTIONS to make ASA more efficient for several kinds of problems (Oliveira, 2001; Oliveira, H.R. Petraglia & Petraglia, 2007; Oliveira, A. Petraglia & Petraglia, 2009). Often, ASA_FUZZY turns on QUENCHing > 1, violating the proof of ASA. For many systems, this speeding up of the sampling process can be a welcome efficiency, but in some systems global minima may be missed. An active research program is to make ASA_FUZZY more adaptive to decreasing as well as increasing QUENCHing.

6. Multiple Systems

Many times hard problems present themselves as multiple systems to be optimized or sampled. Experience shows that all criteria are not always best considered by lumping them all into one cost function, even with some typical methods as Pareto sampling, but rather good judgment should be applied to multiple stages of pre-processing and post-processing when performing such optimization or sampling.

6.1. SELF_OPTIMIZE

The SELF_OPTIMIZE OPTIONS was an early OPTIONS to use ASA itself to optimize parameters used for a particular problem using ASA. A few ASA_TEMPLATEs that come with the code give examples of using SELF_OPTIMIZE.

SELF_OPTIMIZE is not particularly useful as the CPU time is the cross product of the outer-shell using SELF_OPTIMIZE and the inner-shell optimizing the selected problem for each generated state from SELF_OPTIMIZE.

SELF_OPTIMIZE is a recursive algorithm, which may be useful as a guide to sample or optimize other recursive systems. At least, it is demonstrated that ASA is ready for such systems.

6.2. ASA_PARALLEL

For many hard problems, most CPU resources are spent on the cost function calculations, not the overhead of running ASA per se. This knowledge plus the nature of the fat-tailed ASA distribution, which typically gives rise to a high generated state to acceptance state ratio, gave rise to the opportunity to insert hooks for parallel code within ASA, essentially running many generated states in parallel, and then checking for the best acceptance state.

The concept was originally tested on a Connection Machine circa 1990, then in the 1994 National Science Foundation Parallelizing ASA and PATHINT Project (PAPP) mentioned above. It is known to have been used in several industrial settings, including chip design.

6.3. TRD Example of Multiple Systems

The file http://www.ingber.com/asa_examples.txt gives several kinds of use for ASA. An interesting example is in a trading code, Trading in Risk Dimensions (TRD) (Ingber, 2010b). TRD provides examples of both recursive and sequential use of ASA.

There are three levels of optimization/sampling: The section
@@OPTIONAL_DATA_PTR and MULTI_MIN
in http://www.ingber.com/asa_examples.txt gives details and explicit code used in some past versions to demonstrate how this is set up in ASA.

A parameterized trading-rule outer-shell uses the global optimization code Adaptive Simulated Annealing (ASA) to fit parameters of the trading system, e.g., trading rules and trading indicators, to historical data. This is necessary during a Training phase with in-sample data.

A simple fitting algorithm, sometimes requiring ASA, is used for an inner-shell fit of incoming market data to real-world probability distributions. The cost function is typically a simple parameterized exponential distribution representing observed fat-tailed distribution.

A risk-management middle-shell develops portfolio level distributions of copula transformed multivariate distributions (with constituent markets possessing typically different marginal distributions in returns space), generated by Monte Carlo samplings. This The copula code essentially transforms different real-world market distributions into a common multivariate Gaussian space where it makes sense to calculate correlations. There are inverse transformations to come back to individual distributions as needed for some trading indicators. ASA is used to importance-sample weightings (contract sizes) of these markets.

Together with the outer-shell optimization, both the middle-shell portfolio sampling and the inner-shell market distribution fits are processed in Training of in-sample data, Testing of out-of-sample data, e.g., using walk-forward scripts, and during Real-Time trading of incoming market data. This means that during Training, there are recursive uses of ASA: For example, for each generated state of trading-rule and trading-indicator parameters in the outer-shell cost function, ASA is used for both middle-shell and inner-shell optimizations and sampling.

During Testing and Real-Time, after the Training stage has determined a set of best (or sets of good parameters to be post-processed using different technical or fundamental criteria by a different ASA cost function, e.g., during walk forwards), the outer-shell parameters the middle-shell and inner-shell cost functions are run sequentially using their cost functions.

ASA can process these multiple cost functions, using a top-level function to set the OPTIONAL_DATA_PTR OPTIONS to information required to set up each level of optimization.

ASA gives ASA_TEMPLATEs in asa_usr.c to process all these OPTIONS:

If the Pre-Compile Option OPTIONAL_DATA_PTR is set to TRUE, an additional Program Option pointer, Asa_Data_Ptr, becomes available to define an array, of type OPTIONAL_PTR_TYPE defined by the user, which can be used to pass arbitrary arrays or structures to the user module from the asa module. This information communicates with the asa module, and memory must be allocated for it in the user module.

For example, struct DATA might contain an array data[10] to be used in the cost_function. Asa_Data_Dim_Ptr might have a value 2. Set OPTIONAL_PTR_TYPE to DATA. Then, data[3] in struct Asa_Data_Ptr[1] could be set and accessed as Asa_Data_Ptr[1].data[3] in the cost function.

For example, your main program that calls asa_main() would have developed a struct SelectedType *SelectedPointer, and you can call asa_main (SelectedPointer, ...). In asa_usr_asa.h, you would have OPTIONAL_PTR_TYPE set to SelectedType. In asa_usr.c (and asa_usr.h) you would develop asa_main (OPTIONAL_PTR_TYPE *OptionalPointer, ...) and, close to the appropriate ASA_TEMPLATE, you would set Asa_Data_Ptr to OptionalPointer. See the ASA_TEMPLATE in asa_usr.c.

I realize this may sound complex, but with the example provided in
http://www.ingber.com/asa_examples.txt
all this work is fairly easy to implement.

7. Conclusion

A sampling of theory, practical considerations, and experience gained from many users over many years, has produced the current ASA code. If you are “lucky” then a simple entry into the code, e.g., just using the asa_opt file to control some OPTIONS, may do very well for you. However, to keep the ASA code robust for many classes of hard problems, there are many OPTIONS available to properly tune your system to provide a valuable optimization or sampling algorithm.

References

Atiya, A.F., Parlos, A.G. & Ingber, L. (2003) A reinforcement learning method based on adaptive simulated annealing, In: Proceedings International Midwest Symposium on Circuits and Systems (MWCAS), December 2003, IEEE CAS. [URL http://www.ingber.com/asa03_reinforce.pdf]

Binder, K. & Stauffer, D. (1985) A simple introduction to Monte Carlo simulations and some specialized topics, In: Applications of the Monte Carlo Method in Statistical Physics, ed. K. Binder. Springer-Verlag, 1-36.

Bowman, M. & Ingber, L. (1997) Canonical momenta of nonlinear combat, In: Proceedings of the 1997 Simulation Multi-Conference, 6-10 April 1997, Atlanta, GA, Society for Computer Simulation. [URL http://www.ingber.com/combat97_cmi.pdf]

Cerny, V. (1982) A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. Report. Comenius University.

Charnes, A. & Wolfe, M. (1989) Extended Pincus theorems and convergence of simulated annealing. International Journal Systems Science. 20(8), 1521-1533.

Collins, N.E., Egelese, R.W. & Golden, B.L. (1988) Simulated annealing — an annotated bibliography. American Journal Mathematical Management Science. 8(3 & 4), 209-307.

Gelfand, S.B. (1987) Analysis of simulated annealing type algorithms. Ph.D. Thesis. MIT.

Geman, S. & Geman, D. (1984) Stochastic relaxation, Gibbs distribution and the Bayesian restoration in images. IEEE Transactions Pattern Analysis Machine Intelligence. 6(6), 721-741.

Ingber, L. (1989) Very fast simulated re-annealing. Mathematical Computer Modelling. 12(8), 967-973. [URL http://www.ingber.com/asa89_vfsr.pdf]

Ingber, L. (1990) Statistical mechanical aids to calculating term structure models. Physical Review A. 42(12), 7057-7064. [URL http://www.ingber.com/markets90_interest.pdf]

Ingber, L. (1991) Statistical mechanics of neocortical interactions: A scaling paradigm applied to electroencephalography. Physical Review A. 44(6), 4017-4060. [URL http://www.ingber.com/smni91_eeg.pdf]

Ingber, L. (1992) Generic mesoscopic neural networks based on statistical mechanics of neocortical interactions. Physical Review A. 45(4), R2183-R2186. [URL http://www.ingber.com/smni92_mnn.pdf]

Ingber, L. (1993a) Adaptive Simulated Annealing (ASA). Global optimization C-code. Caltech Alumni Association. [URL http://www.ingber.com/#ASA-CODE]

Ingber, L. (1993b) Simulated annealing: Practice versus theory. Mathematical Computer Modelling. 18(11), 29-57. [URL http://www.ingber.com/asa93_sapvt.pdf]

Ingber, L. (1993c) Statistical mechanics of combat and extensions, In: Toward a Science of Command, Control, and Communications, ed. C. Jones. American Institute of Aeronautics and Astronautics, 117-149. [ISBN 1-56347-068-3. URL http://www.ingber.com/combat93_c3sci.pdf]

Ingber, L. (1996a) Adaptive simulated annealing (ASA): Lessons learned. Control and Cybernetics. 25(1), 33-54. [Invited paper to Control and Cybernetics on "Simulated Annealing Applied to Combinatorial Optimization." URL http://www.ingber.com/asa96_lessons.pdf]

Ingber, L. (1996b) Canonical momenta indicators of financial markets and neocortical EEG, In: Progress in Neural Information Processing, ed. S.-I. Amari, L. Xu, I. King & K.-S. Leung. Springer, 777-784. [Invited paper to the 1996 International Conference on Neural Information Processing (ICONIP'96), Hong Kong, 24-27 September 1996. ISBN 981 3083-05-0. URL http://www.ingber.com/markets96_momenta.pdf]

Ingber, L. (1996c) Statistical mechanics of nonlinear nonequilibrium financial markets: Applications to optimized trading. Mathematical Computer Modelling. 23(7), 101-121. [URL http://www.ingber.com/markets96_trading.pdf]

Ingber, L. (1997) Statistical mechanics of neocortical interactions: Applications of canonical momenta indicators to electroencephalography. Physical Review E. 55(4), 4578-4593. [URL http://www.ingber.com/smni97_cmi.pdf]

Ingber, L. (1998a) Data mining and knowledge discovery via statistical mechanics in nonlinear stochastic systems. Mathematical Computer Modelling. 27(3), 9-31. [URL http://www.ingber.com/path98_datamining.pdf]

Ingber, L. (1998b) Statistical mechanics of neocortical interactions: Training and testing canonical momenta indicators of EEG. Mathematical Computer Modelling. 27(3), 33-64. [URL http://www.ingber.com/smni98_cmi_test.pdf]

Ingber, L. (2001a) Adaptive Simulated Annealing (ASA) and Path-Integral (PATHINT) Algorithms: Generic Tools for Complex Systems. ASA-PATHINT Lecture Plates. Lester Ingber Research. [Invited talk U Calgary, Canada, April 2001. URL http://www.ingber.com/asa01_lecture.pdf and asa01_lecture.html]

Ingber, L. (2001b) Statistical Mechanics of Combat (SMC): Mathematical Comparison of Computer Models to Exercise Data. SMC Lecture Plates. Lester Ingber Research. [URL http://www.ingber.com/combat01_lecture.pdf and combat01_lecture.html]

Ingber, L. (2001c) Statistical Mechanics of Financial Markets (SMFM): Applications to Trading Indicators and Options. SMFM Lecture Plates. Lester Ingber Research. [Invited talk U Calgary, Canada, April 2001. Invited talk U Florida, Gainesville, April 2002. Invited talk Tulane U, New Orleans, January 2003. URL http://www.ingber.com/markets01_lecture.pdf and markets01_lecture.html]

Ingber, L. (2001d) Statistical Mechanics of Neocortical Interactions (SMNI): Multiple Scales of Short-Term Memory and EEG Phenomena. SMNI Lecture Plates. Lester Ingber Research. [Invited talk U Calgary, Canada, April 2001. URL http://www.ingber.com/smni01_lecture.pdf and smni01_lecture.html]

Ingber, L. (2005) Trading in Risk Dimensions (TRD). Report 2005:TRD. Lester Ingber Research. [URL http://www.ingber.com/markets05_trd.pdf]

Ingber, L. (2006) Ideas by statistical mechanics (ISM). Report 2006:ISM. Lester Ingber Research. [URL http://www.ingber.com/smni06_ism.pdf]

Ingber, L. (2007a) Ideas by Statistical Mechanics (ISM). Journal Integrated Systems Design and Process Science. 11(3), 31-54. [Special Issue: Biologically Inspired Computing.]

Ingber, L. (2007b) Real Options for Project Schedules (ROPS). Report 2007:ROPS. Lester Ingber Research. [URL http://www.ingber.com/markets07_rops.pdf]

Ingber, L. (2008a) AI and Ideas by Statistical Mechanics (ISM), In: Encyclopedia of Artificial Intelligence, ed. J.R. Rabuñal, J. Dorado & A.P. Pazos. Information Science Reference, 58-64. [ISBN 978-1-59904-849-9]

Ingber, L. (2008b) Statistical mechanics of neocortical interactions (SMNI): Testing theories with multiple imaging data. NeuroQuantology Journal. 6(2), 97-104. [URL Invited paper. http://www.ingber.com/smni08_tt.pdf]

Ingber, L. (2009) Statistical mechanics of neocortical interactions: Nonlinear columnar electroencephalography. NeuroQuantology Journal. 7(4), 500-529. [URL http://www.ingber.com/smni09_nonlin_column_eeg.pdf]

Ingber, L. (2010a) Real Options for Project Schedules (ROPS). International Journal of Science, Technology & Management. 2(2), 15-20. [Invited paper]

Ingber, L. (2010b) Trading in Risk Dimensions, In: The Handbook of Trading: Strategies for Navigating and Profiting from Currency, Bond, and Stock Markets, ed. G.N. Gregoriou. McGraw-Hill, 287-300.

Ingber, L., Fujio, H. & Wehner, M.F. (1991) Mathematical comparison of combat computer models to exercise data. Mathematical Computer Modelling. 15(1), 65-90. [URL http://www.ingber.com/combat91_data.pdf]

Ingber, L. & Mondescu, R.P. (2001) Optimization of trading physics models of markets. IEEE Trans. Neural Networks. 12(4), 776-790. [Invited paper for special issue on Neural Networks in Financial Engineering. URL http://www.ingber.com/markets01_optim_trading.pdf]

Ingber, L. & Mondescu, R.P. (2003) Automated internet trading based on optimized physics models of markets, In: Intelligent Internet-Based Information Processing Systems, ed. R.J. Howlett, N.S. Ichalkaranje, L.C. Jain & G. Tonfoni. World Scientific, 305-356. [Invited paper. URL http://www.ingber.com/markets03_automated.pdf]

Ingber, L. & Sworder, D.D. (1991) Statistical mechanics of combat with human factors. Mathematical Computer Modelling. 15(11), 99-127. [URL http://www.ingber.com/combat91_human.pdf]

Johnson, D.S., Aragon, C.R., McGeoch, L.A. & Schevon, C. (1987) Optimization by simulated annealing: An experimental evaluation (Parts 1 and 2). Report. AT&T Bell Laboratories.

Kirkpatrick, S., Gelatt, C.D., Jr. & Vecchi, M.P. (1983) Optimization by simulated annealing. Science. 220(4598), 671-680.

Ma, S.-K. (1985) Statistical Mechanics. World Scientific, Philadelphia.

Mathews, J. & Walker, R.L. (1970) Mathematical Methods of Physics, 2nd ed.. Benjamin, New York, NY.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. & Teller, E. (1953) Equation of state calculations by fast computing machines. Journal of Chemical Physics. 21(6), 1087-1092.

Oliveira, H.A., Jr. (2001) Fuzzy control of stochastic global optimization algorithms and very fast simulated reannealing. Report. hime@engineer.com. [URL http://www.optimization-online.org/DB_FILE/2003/11/779.pdf]

Oliveira, H.A., Jr., Petraglia, A. & Petraglia, M.R. (2009) Frequency domain FIR filter design using fuzzy adaptive simulated annealing. Circuits, Systems, and Signal Processiing. 28(6), 899-911. [DOI: 10.1007/s00034-009-9128-1]

Oliveira, H.A., Jr., Petraglia, H.R. & Petraglia, A. (2007) Frequency domain FIR filter design using fuzzy adaptive simulated annealing, In: 7th International Symposium on Signal Processing and Information Technology, 2007, vol. 1, Proceedings of ISSPIT, 899-903.

Pincus, M. (1970) A Monte Carlo method for the approximate solution of certain types of constrained optimization problems. Operations Research. 18, 1225-1228.

Szu, H. & Hartley, R. (1987) Fast simulated annealing. Physics Letters A. 122(3-4), 157-162.

van Laarhoven, P.J.M. & Aarts, E.H.L. (1987) Simulated Annealing: Theory and Applications. D. Reidel, Dordrecht, The Netherlands.