How Low Can You Sink?

In Search of Global Minima

Vivek S Borkar

Vivek S Borkar obtained his Ph D from the University of California, Berkeley, in 1980 and is now with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. He likes to work on theoretical problems with an interesting component of randomness or interesting dynamics, preferably both.

Many scientific and technological problems involve finding the lowest attainable value of a function. This article motivates, describes, and discusses one powerful computational strategy. It is called "simulated annealing" in analogy with the physical process of slowly cooling a system to take it to its ground state.

Getting to the Bottom of It

There are many situations where one wants to maximize or minimize something. For example, one may want to maximize returns on an investment or minimize losses, or, as an engineer, maximize energy efficiency or minimize estimation errors. Isolating the underlying mathematical problem, one then has a map f from a given domain D to the real line and the problem is to find the point in D (if any) where f attains its maximum or minimum. It helps to consider D as being laid out *horizontally* (takes some imagination if, say, D = the n-dimensional vector space with $n \geq 3$) and for each x in D, $f(x)$ plotted *vertically* to give a graph of $f(x)$ vs x. Assuming f to be continuous, the graph can be visualized as a landscape with peaks and valleys. The peaks will correspond to *local maxima* i.e., points where f is the maximum with respect to its immediate neighbourhood. The bottoms of the valleys are then the local minima, defined analogously. For the sake of being specific, consider the minimization problem. The aim then is to find the global minimum, the bottom of the lowest valley.

One may want to maximize returns on an investment or minimize losses, or, as an engineer, maximize energy efficiency or minimize estimation errors.

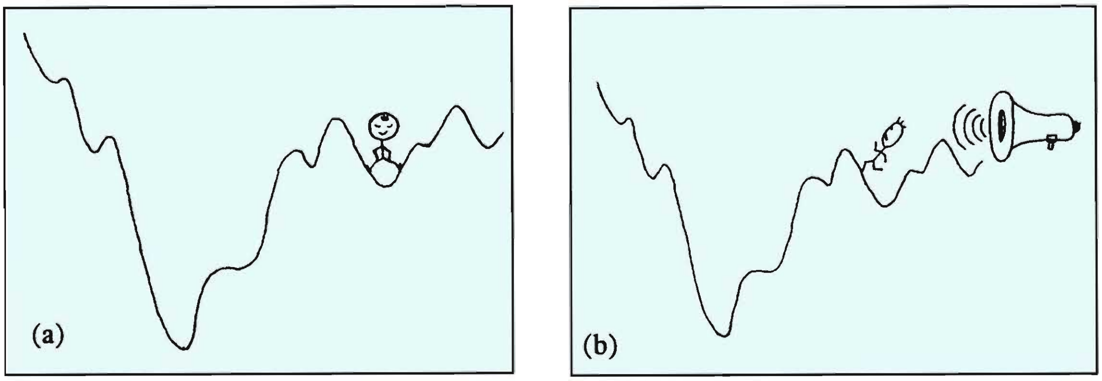
The obvious thing to do is to keep going downhill as long as you can. This is what *descent* algorithms do. Many popular algorithms (steepest descent, conjugate gradient) are of this variety.

Even the other rival class, that of Newton and quasi-Newton algorithms, can be construed as *descents* modulo some mental gymnastics that allows the concept of distance to vary from point to point. (The buzz word here is *variable metric*).

But these algorithms, which are deterministic (that is, non-random) and use only local information, can't tell one local minimum from another and can get trapped in one that is far from the best. How then, is one to find the global minimum? A naive response would be to 'get a bird's eye view of the landscape, spot a global minimum and make a bee-line for it'. But algorithms cannot do what the birds and the bees can, their proliferation notwithstanding. For any optimization problem worth its salt, using global information is not easy. So one attacks the other flank of these algorithms, their determinism. The trick is to add some random noise to the algorithm which, whenever the algorithm threatens to sit pretty in a local minimum (*Figure 1a*), gives it a gentle uphill push (*Figure 1b*). But if you keep adding noise, you will be marching the algorithm up and down the hills forever like the proverbial grand old Duke of York. So one has to reduce the noise level slowly, to attain a balance between random *exploration* of the landscape a la a pure random walk, and *exploitation* of the gradient (slope) information as in descent methods. The algorithm described in this article, called *simulated annealing*, does precisely that.

A naive response would be to 'get a bird's eye view of the landscape, spot a global minimum and make a bee-line for it'. But algorithms cannot do what the birds and the bees can.

Figure 1 a, b.



Fate of a Salesman

A salesman has to visit N cities, each of them exactly once, and return to where he started. He knows the pairwise distances between these cities and his problem is to find the best sequence in which the cities should be visited so as to minimize the total distance travelled.

Before plunging into the mathematics of the algorithm, we take a look at an archetypical problem where this algorithm has been applied. This is the celebrated *travelling salesman* problem. A salesman has to visit N cities, each of them exactly once, and return to where he started. He knows the pairwise distances between these cities and his problem is to find the best sequence in which the cities should be visited so as to minimize the total distance travelled.

Each possible ordering of the cities involving each city exactly once is called a tour. To each tour one assigns a cost, a positive number equal to the total distance the salesman would travel if he were to visit the cities in that order. The problem then is to minimize this cost over the finite set of tours.

This may sound simple. (After all, there are only finitely many tours!) But it is not. The number of possible tours grows explosively with N (as $N!$). This rules out simple search algorithms. Things still wouldn't be too bad if there was enough 'structure' to the problem, but there isn't. The problem is provably hard, i.e., it belongs to an equivalence class of problems known to be hard in a precise technical sense.

A popular heuristic algorithm for the travelling salesman problem is the *2-opt*. Here one starts with a tour and randomly picks two consecutive cities on the tour. They are interchanged to obtain a new tour. If the new tour has a lower cost replace the old tour by the new tour. This procedure is repeated until there is no further improvement.

The problem with this and other similar heuristics is that they get stuck in local minima. If the number of cities is small, say, in tens, they still do reasonably well (in fact, often better than simulated annealing for a fixed finite run length of the algorithms). If N exceeds a few hundreds, however, simulated annealing begins to show a distinct edge over these heuristics.

We shall return to this and related problems later after looking into the formal mathematical aspects of the algorithm. A reader uncomfortable with mathematical probability theory may skip the next section at a first pass.

You Have Nothing to Lose But Your Chains

For simplicity, let D be a finite set with M elements. The mathematical model for our algorithm is a Markov chain. A Markov chain on D is a random process $X_n, n = 0, 1, 2, ...,$ taking values in D with the property that the probability of its moving from i to j at any given time does not depend on how it arrived at i. (Technically speaking, its future and past are conditionally independent given the present, a good philosophy for life in general.) If this probability is also independent of the explicit time count (the clock), we may denote it by $p(i,j)$. If the probability of the chain being in i at a given time is $\pi(i)$, the probability of its being in j at the next instant will be $\sum_i \pi(i) p(i,j)$. Thus if the probability vector $\pi(\cdot)$ satisfies

$$\sum_j \pi(i) p(i,j) = \pi(j), \ j \in D, \tag{1}$$

then we have — if the probability distribution of the chain is $\pi(.)$ at some time, then it is $\pi(.)$ forever. Such a $\pi(.)$ is called a *stationary distribution*. If the chain is irreducible, i.e., can go from any $i \in D$ to any $j \in D$ with positive probability, a unique such $\pi(\cdot)$ exists. Also, the fraction of time the chain spends in i approaches $\pi(i)$ in the limit for each i.

Equation (1) is called *global balance*. (Think of equilibrium concentration at j being equal to that at i times the rate of flow from i to j, summed over i.) One also has the *detailed balance* equation

$$\pi(i) p(i,j) = \pi(j) p(j,i), i, j \in D, \tag{2}$$

That is, if the selected move is downhill, it is made with certainty. If it is uphill, it is made with a small probability which decreases with the amount of climb involved.

which implies (1) but not vice versa. For most chains (2) may not be possible. Mercifully, it is so for the chain we are about to consider.

Suppose we impose on D a neighbourhood structure whereby each $i \in D$ has N neighbours (N usually much smaller than M) and i, j are either neighbours or they are not. Consider a chain which (i) cannot go from i to j unless j is a neighbour of i, (ii) can with probability

$$p(i, j) = (1/N) \exp(-(f(j) - f(i))^{+} / T) \qquad (3)$$

if it is, and (iii) remains in i with the remaining probability. Here $T>0$ is a parameter called *temperature* for reasons that will become apparent later, and $x^{+} = \max(x, 0)$. The expression (3) can be thought of as being in two parts: the 'selection probability' given by $1/N$ and the 'acceptance probability' given by the exponential term. The interpretation is that the chain at i picks a neighbour j with equal probability. It moves there with a probability equal to the acceptance probability and remains at i with the remaining probability. Note that the acceptance probability is 1 if $f(j) < f(i)$[1] and <1 otherwise, decreasing as $f(j)$ increases. That is, if the selected move is downhill, it is made with certainty. If it is uphill, it is made with a small probability which decreases with the amount of climb involved.

[1] This is because the exponent in equation (3) becomes zero, from the definition of the "f" operation given below equation (3).

It is easily verified that given (3), (2) is satisfied by

$$\pi_T(i) = Z^{-1} \exp(-f(i)/T), \quad i \in D,$$

where Z is the normalizing factor. Observe that $f(i) > f(j)$ implies $\pi_T(i) < \pi_T(j)$. Thus $\pi_T(\cdot)$ assigns maximum probability to the i where f attains its minimum. The chain then spends the maximum fraction of time in these states. What's more, the smaller the parameter T, the larger this fraction (the more peaked $\pi_T(.)$ is at the global minimum). Note that the two limiting cases of $T=0$ and $T=\infty$ correspond respectively to pure descent and pure random walk.

To fix ideas, consider the specific case of $D = \{1, 2, 3, 4\}$ with each element a neighbour of every other. Let $f(i) = i$ for i in D and $T = 1$. Then we have (using e=2.71828...) = base of natural logarithms.

$p(4,3) = p(4,2) = p(4,1) = p(3,2) = p(3,1) = p(2,1) = 1/3,$
$p(1,2) = p(2,3) = p(3,4) = 1/(3e)$
$p(1,3) = p(2,4) = 1/(3e^{2)}$
$p(1,4) = 1/(3e^3).$

This uniquely specifies $p(i,i)$ as $1-\sum_{j \neq i} p(i,j)$ for each i. In view of the foregoing, one has

$$\pi_T(i) = e^{-i}/(e^{-1}+e^{-2}+e^{-3}+e^{-4}),$$

which clearly peaks at $i=1$, the global minimizer of f.

The foregoing suggests the following algorithm. Run the chain with a time-varying $T=T(t)$. Decrease $T(t)$ with time t so slowly that the probability distribution of the chain closely tracks $\pi_{T(t)}$, thus concentrating on the global minimum in the limit. This is the *simulated annealing* algorithm, named thus by analogy with the eponymous slow cooling process for hardening metals. The function $t \to T(t)$ is correspondingly called the cooling schedule.

That is, the probability of the chain being away from the set S of global minima goes to zero. This does not ensure that with probability one, the algorithm hits S and stays there.

In the Long Run

But does the algorithm work provably? Yes, if $T(t)$ decreases slowly enough to ensure $\Sigma_t \exp(-d/T(t)) = \infty$ where d = the 'depth' of the problem. This is defined as the maximum of the minimum one has to climb from any point in D in order to get to some global minimum. The convergence, however, is only in probability. That is, the probability of the chain being away from the set S of global minima goes to zero. This does not ensure that with probability one, the algorithm hits S and stays there. That would be *almost sure convergence* in probabilistic jargon, a stronger concept. In both, the probability of the set of *bad* sample points (i.e., the set A_n on which $X_n \notin S$) shrinks to zero. But in the

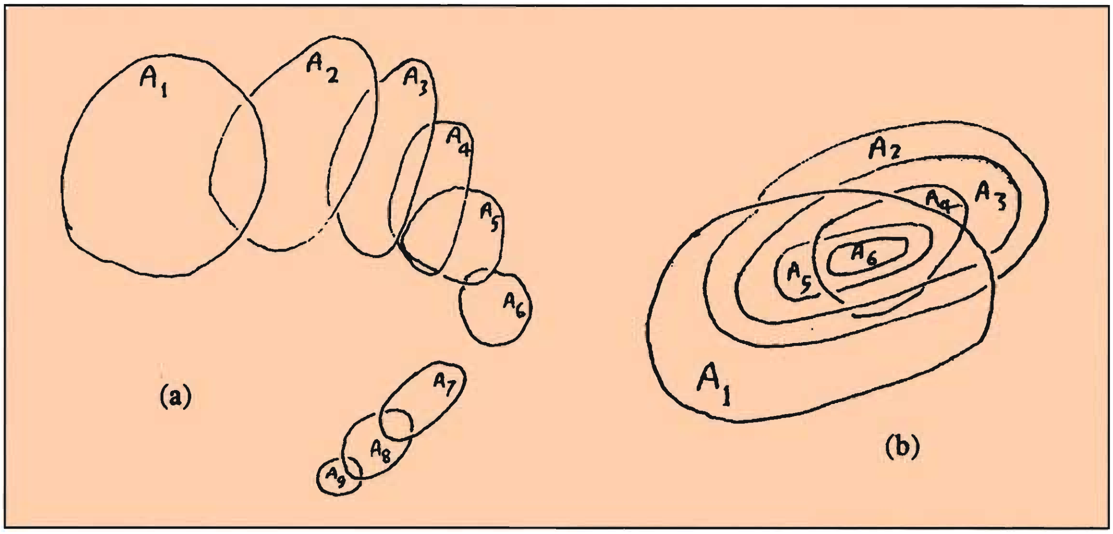


Figure 2 a,b.

former, these can wander all over the sample space while shrinking (*Figure 2a*), unlike the latter, where they do so in a more or less nested fashion (*Figure 2b*). (The outer square in these figures represents the underlying sample space.) Unfortunately, almost sure convergence to S need not hold. One can concoct situations wherein the algorithm leaves S infinitely often, though such occurrences become increasingly rare. But the fraction of time spent in S does go to unity.

To get a feel for this rather subtle point, consider the sequence $\{a(n)\}$ which is 1 when $n = 2^m$ for some $m \geq 1$, zero otherwise. It becomes 1 increasingly rarely and the fraction of time it spends in zero approaches one, though it does not converge to zero.

Intuitively, what is required to make the algorithm stick to a global minimum is that the immediate valley surrounding it be deep.

While commenting on what happens in the long run, the economist John Maynard Keynes once quipped that in the long run, we are all dead.

But these are only asymptotic results. While commenting on what happens in the long run, the economist John Maynard Keynes once quipped that in the long run, we are all dead. A similar cynicism may be warranted here in the absence of good rate of convergence results. Few analytic results are available, such as some that predict inverse polynomial (in t) decay of the probability of not hitting S until time t. But the constants up front involved in these estimates are too large to make them practically useful. One thus has to fall back on empirical observations, which are encouraging for certain classes of problems. We shall discuss these in the last section.

From Child's Play to Statistical Mechanics

We now present two interpretations of the algorithm. The first is simply an analogy with a children's toy wherein a metal ball in a maze is to be pushed to its center by repeatedly tilting the maze in a suitable manner. One starts out with rather large tilts, and then smaller ones as the ball approaches the center. The addition of slowly diminishing noise to the basic descent scheme can also be thought of as randomly tilting the landscape, the extent of which diminishes with time. The algorithm trapped in a local minimum is thus *poured out* by tilting the landscape. This analogy is not as far fetched as it may seem. It is quite accurate for simulated annealing in a finite dimensional vector space (as opposed to a finite set D) which, unfortunately, we will not consider here because of its technicalities.

The first is simply an analogy with a children's toy wherein a metal ball in a maze is to be pushed to its center by repeatedly tilting the maze in a suitable manner. One starts out with rather large tilts, and then smaller ones as the ball approaches the center.

The second interpretation runs deeper and in fact, motivated the *Monte Carlo* Markov chains that preceded simulated annealing. (They correspond to the constant T version thereof.) Recall that a thermodynamic system in equilibrium at a constant temperature T minimizes its (Helmholtz) free energy, which is its internal energy minus T times its entropy. Statistical mechanics, which aims to derive thermodynamics from microscopic phenomena, translates this into the following: Let D be a discrete set of possible states and $f(i)$ the energy in state $i \in D$. If p_i is the probability of the system being in state i, $\sum_i p_i f(i)$ is the average energy. On the other hand, the entropy of $p = [p_1, \ldots, p_m]$ is given by $-\sum_i p_i \ln p_i$, *its information content.* This can be justified *axiomatically* and the readers unfamiliar with information theory are requested to accept it on faith. The free energy minimization principle then requires p to minimize $\sum_i p_i\ (f(i)+T\ln p_i\ )$ subject to $\sum_i p_i=1$. This is a strictly convex function (i.e., function with the property that the line joining any two points on its graph lies above the graph) on a bounded convex domain (i.e., a set that contains every line segment whose end points are in the set.) This makes it an optimization theorist's dream problem. It has a unique solution given by $p=\pi_T$. In the $T\to 0$ limit, the problem reduces to minimizing $\sum_i p_i f(i)$, which is tantamount to our original problem. (Think about it! Strictly speaking, it is a 'relaxation' of the original problem in optimization parlance.) The algorithm thus simulates convexification of the problem, with lowering of T corresponding to gradually distorting the convex problem to the original. There are optimization techniques called *homotopy* methods which actually do this. The difference here is that the convexification is not explicit. It arises through the average behavior of a random phenomenon.

One may then ask: Why not do the deterministic minimization of free energy directly? The reason is that in most applications, D is very large and complex, the space of probability vectors on D even more so. Thus the deterministic problem is not usually computationally amenable. There are, however, applications (like image processing) where the above considerations have led to deterministic approximations of simulated annealing. These are called mean field annealing methods, after the 'mean field' theories of physics wherein one replaces fluctuating quantities by their averages.

The Good, the Bad, and the Ugly

When should one use simulated annealing? The following considerations give some intuition about this. Consider successive blocks of a fixed, large number of iterations. The higher the value of T, the more the algorithm will wander in any such block. Thus at high T, it sees the landscape on a coarse scale, seeking only the broad valleys. As T is lowered, it starts seeing finer length scales and hence smaller valleys.

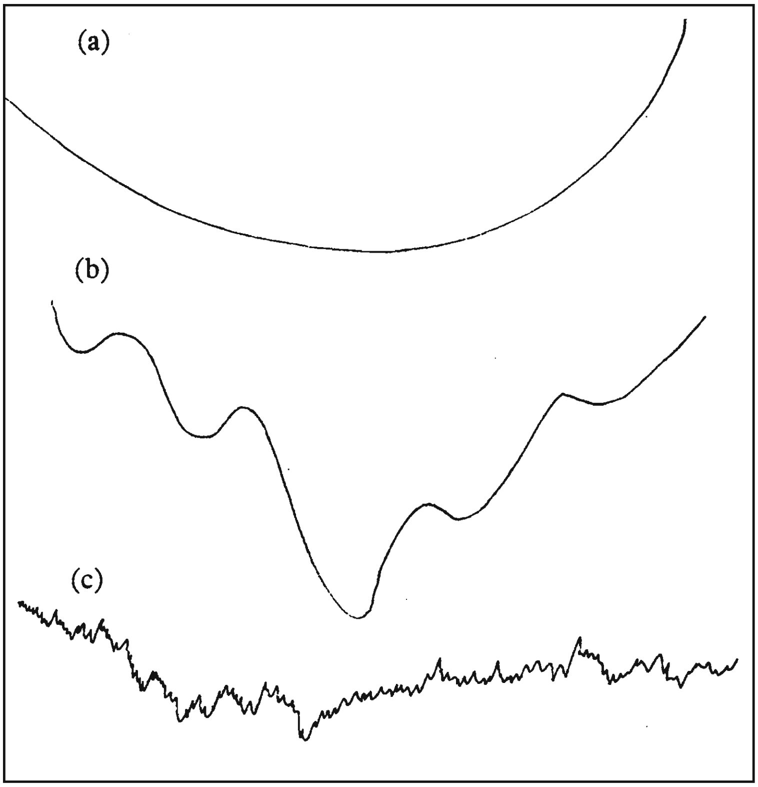


Figure 3 a–c.

Keeping this in mind, consider the three functions displayed in *Figure 3*. The first is a *good* function, smooth, convex, with a unique local-cum-global minimum. Any reasonable algorithm will work for this. The second function is *bad*, but not altogether so. Reasonable heuristics like the multistart method (which initiates several descent algorithms at many randomly chosen initial conditions) will do well on this with a high probability. This is more than what one can say for the third function, which is really *ugly*. It has local minima at all length scales, making it tough for even the most reasonable of the traditional heuristics. But when things get tough, the tough get going and simulated annealing, which is *intelligent brute force* by another, is a tough algorithm. In fact, it is only the large and ugly problems for which it starts showing significant gains.

But do we encounter such ugly problems in practice? Plenty! Combinatorial optimization is a real gold mine of these. (Perhaps 'minefield' would be a better metaphor.) Examples are: graph matching, graph partitioning, graph colouring, and travelling salesman problems. In fact, the algorithm was originally introduced for solving combinatorial optimization problems in VLSI circuit design.

Another major area is image processing, where a noisy image (sometimes misleadingly called a dirty picture) is cleaned by optimally fitting to it a nicer image with respect to some error criterion.

The reader might have heard of neural networks which are large networks of simple nonlinear elements whose parameters are adjusted to perform specific tasks like associative memory or pattern classification. One of these is the *Boltzmann machine*, which uses simulated annealing for optimal parameter adjustments.

Condensed matter physicists encounter complex systems called *spin glasses* with really ugly energy functions. Simulated annealing is a useful tool for numerically analyzing these.

The above problems have an important feature in common, which we illustrate in the case of the travelling salesman problem described in an earlier section. D then is the set of all possible tours. Two tours are neighbours if one is obtainable from the other by interchanging the placing of two cities that occur successively. For large N, the tour length is difficult to compute, but the difference in tour length of two neighbours is not. That is, $f(i)$ is hard to find, but $f(j) - f(i)$ is not when i, j are neighbours. This is another important feature of typical application domains of simulated annealing. In fact, if it were not so, simple random walk would do better simply by keeping track of the lowest point visited so far.

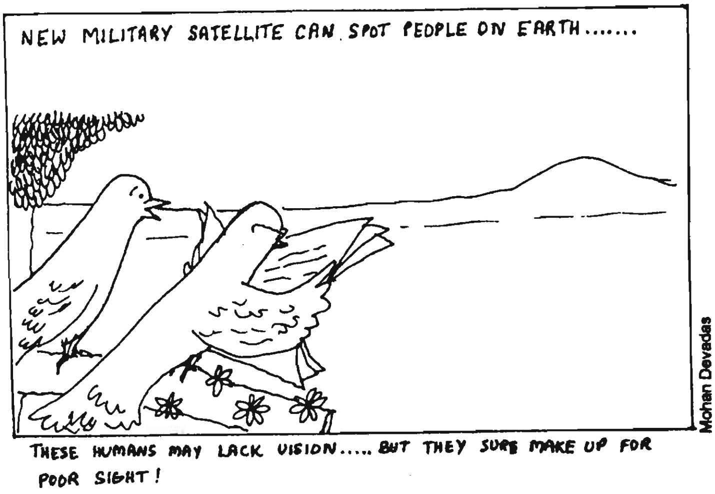
Finally, the remarks at the beginning of this section also give a clue as to what T to start with. If T is too high, it is virtually a random walk and we are wasting resources. If too low, one may take forever to move out of the current valley. As T decreases from *high* to *low*, one expects π_T to go from an almost flat distribution to a humped one. In many cases, the transition is fairly sharp around a *critical temperature* T_c. The rule of thumb is to use T_c as the initial T. Of course, T_c has to be guessed or estimated, which is another problem altogether.

In practice, of course, there are many ad hoc add-on features to speed-up the algorithm or reduce its resource requirements, usually at the expense of exact optimality. After all, there is a science of optimization and there is also an art to it. It is the former that is being conveyed to you in this article. The latter cannot be, since one has to simply 'grow into it' through experience.

Suggested Reading

- P Van Laarhoven and E Aarts. *Simulated Annealing: Theory and Applications*. D Reidel. Dordrecht, 1987.
- D Bertsimas and J Tsitsiklis. Simulated Annealing. *Probability and Algorithms*. National Academy Press. Washington D.C., 1992.

Address for Correspondence
Vivek S Borkar
Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore 560 012, India
email: borkar@cas.iisc.ernet.in


NEW MILITARY SATELLITE CAN SPOT PEOPLE ON EARTH.......
THESE HUMANS MAY LACK VISION..... BUT THEY SURE MAKE UP FOR POOR SIGHT!
Mohan Devadas