Stochastic Search Methods

Sanza T. Kazadi

11 February 1995

1 Introduction

Many problems in science require that a multidimensional vector be found which can presumably satisfy the constraints placed on a particular physical system and provide an optimal solution to the problem. In principle this is a difficult problem. If the dimension of the problem is much larger than three, it is difficult, if not impossible to guess the solution or find it by random search. As a result, several techniques have been created to solve these problems. These techniques are stochastic methods, and all have elements which relate to hillclimbing methods.

The basic premise behind a stochastic method is the following. In direct hillclimbing methods, problems with 'bumpy' fitness landscapes pose serious threats to the usefulness of the technique. The plethora of local minima and maxima can trap an iterative search, ending the search in a suboptimal search. Problems like this can be alleviated to some extent by increasing the step size, but this can often times lead to imprecise final answers. Moreover, different step sizes can often lead the search along different paths, effecting the final outcomes. To deal with these problems, stochastic methods were developed. Stochastic methods take particular steps in the n-dimensional space which do not necessarily lead to an increase in overall fitness of the current solution or solution set. In this way, the stochastic methods can often times avoid these local minima problems, to find the global maxima. On the other hand, these methods are limited by many of the same things that limit hillclimbing methods.

In the following pages, we will present a set of stochastic search methods. While it can be shown that some of these search methods are actually identical, we will ignore this. We will attempt to show the reasons for using population-based search paradigms in some cases, and for single-solution paradigms in other cases.

2 Directed Random Search

2.1 Definition

There are many different ways of searching an n-dimensional vector[1] space. The most

[1]In this paper, we will interchangably use the words vectors and solutions.

direct way is a random search. In this paradigm, random vectors are evaluated, with the highest scoring vector being retained as the search result. This particular search has a few problems. In finite spaces, this is a reasonable way of finding a solution if we assume that the function varies smoothly with the vectors. If we are searching for a good solution, we need only randomly find a 'good' vector, and then use a hillclimbing method to find the absolute optimum. On the other hand, this is not a good solution if we have an unbounded space, and, moreover, cannot narrow our search by problem-specific reasoning.

One direct solution to this problem which solves our problem of unbounded solutions is the *directed random search.* In this method, we begin with a particular vector, randomly chosen. Next, we choose several random vectors located in an n-ball of radius $\epsilon > 0$ and with the center at the chosen vector. All of these vectors are evaluated, and the one with the highest function value is stored. All others are discarded. The process is iterated until no higher values are found. Variations on this theme allow for the n-ball to be increased in size once the solution has apparently been found, and this allows the method to jump over larger hills.

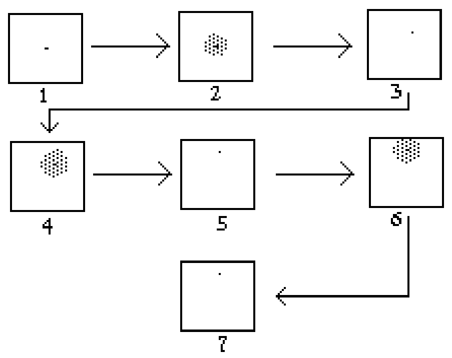
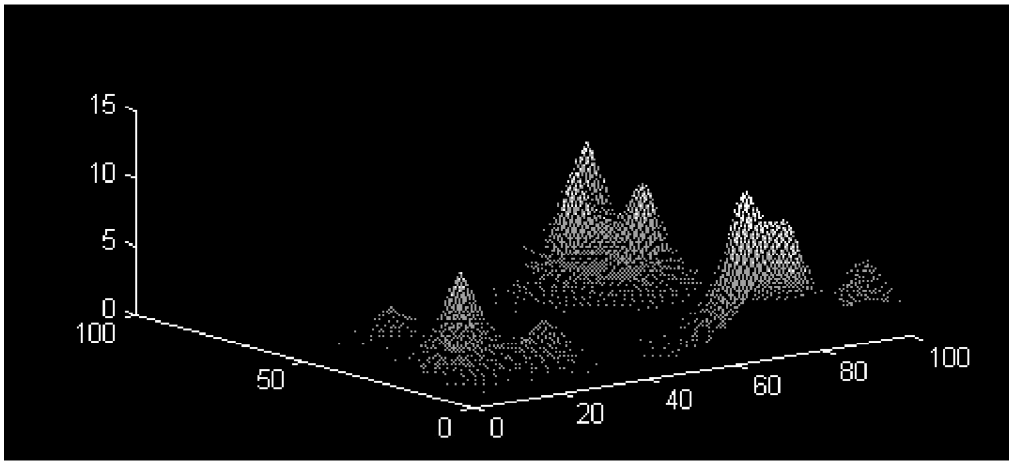


Figure 1: A typical Directed Random Search

2.2 Strengths and Weaknesses of the Method

Let us try to understand when this method will work, and when it won't. First, we need a geometric idea of what is going on. In our case, we start with a point. By choosing an $\epsilon > 0$, we are restricting the search to the space directly around the point. That means that if we are at a point located on a local maxima, and the 'next hill' is a greater distance than ϵ from the point, we will never find the other hills. Thus, the directed random search is very dependant on local properties of the function. While it will robustly climb hills, and jump over small or narrow maxima, it will not climb down extended peaks.

On the other hand, let us suppose that the fitness landscape is made up of several narrow peaks.



Now, again, if the step size is too small, then the algorithm will tend to get trapped on whatever peak it initially climbs. On the other hand, if the stepsize is very large, then the algorithm will have to densely cover the $\epsilon-$ball in order to find a peak correctly. Thus, the algorithm is highly dependant on the step size and initial conditions. Moreover, it has no mechanism for back-tracking. While it can jump over small peaks, it cannot climb down peaks in order to find other higher peaks.

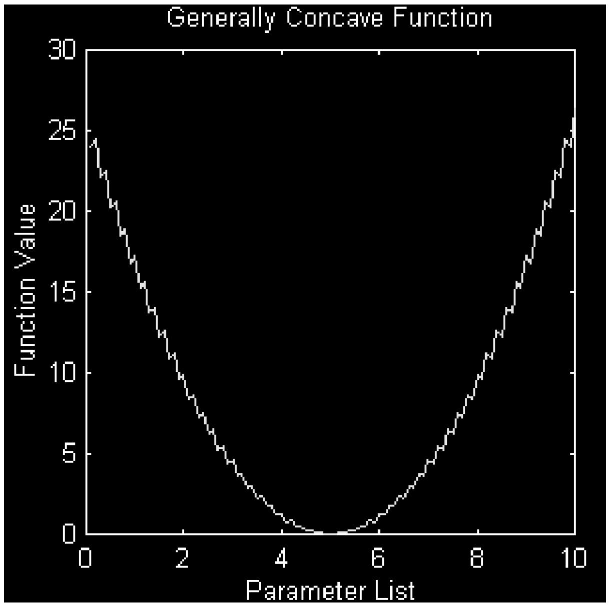
This technique has several attractive qualities. The obvious one is that the technique is cheap. It is easily implemented and is fast. Moreover, by varying the starting positions, the final positions for more complicated function landscapes will be varied, and this will tend to reflect the complexity of the function. Knowing the complexity of the function can help to decide what more advanced methods to use. Finally, the technique can be very precise. Once a peak has been found, refinements made to the original solution only serve to improve the precision. If the technique finds a good solution, this solution will be very close to the exact solution.

In using this technique, it is important to know what function you are using. This technique is useful if it recovers the same result from many different starting positions, and using many different choices of ϵ. It is not useful if these conditions do not hold.

3 Simulated Annealing

3.1 Definition

The second method typically used is a single-element search called *simulated annealing.* This technique is an abstraction of the spinglass problem in physics. The idea is to allow a number of quantum states of a particular system, allowing the system to 'bounce around' to many different states, eventually settling on a final state. In spinglasses, this is allowed because the final low-energy state is very ordered, and the function is generally concave (meaning a concave function punctuated by many small peaks).

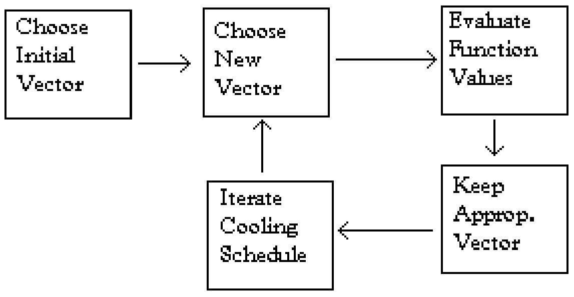


Simulated annealing is defined in the following way. Suppose that we have a vector space Γ and we choose a vector $\nu \in \Gamma$. We also suppose that there is a function which we are trying to optimize (known in physics as the *Hamiltonian*, or the *energy function*) f which maps Γ to the real numbers. Then, we also choose a temperature $\tau > 0$. Once this temperature is chosen, we may choose a new vector $\nu' \in \Gamma$ which is related to ν by adding a small perturbation to ν to obtain ν'. Now, we may keep this new vector with probability

$$p = e^{\frac{f(\nu') - f(\nu)}{\tau}} \tag{1}$$

where a probability exceeding 1 is interepreted as an automatic acceptance. If the temperature is very small, than any new vector scoring less than the old vector is immediately rejected, while high temperatures are very likely to accept it.

It is easy to see why this might work for large temperatures. Small hills such as those given in the preceding figure are easily overcome, and the function may easily move down the general energy gradient. However, convergence requires that the acceptance of poor solutions decreases as the simulation continues. This requires that we introduce a *cooling schedule* whereby the 'temperature' τ slowly decreases. In generally concave functions, this means that the function will converge to the true solution rather directly, providing that the cooling schedule is sufficiently slow.



Simulated Annealing General Flow Chart

3.2 Strengths and Weaknesses of the Method

This method has several strengths. The first, and perhaps the most important, is its ability to walk over and around peaks of different heights and of different widths with the same stepsize. This technique is used in the Metropolis et al. Monte Carlo (for entirely different reasons), and can be used to simulate quantum tunnelling, perhaps for obvious reasons. The method is also quite robust. With the correct mutation capabilities, the function can reach many peaks from one starting position, and thus can avoid being limited by its starting position. This means that it may start on one hill, and eventually climb an adjacent hill. In fact, it may even climb down the other side of that adjacent hill and climb up another hill.

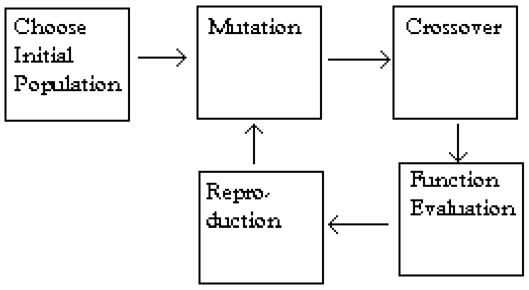
The fundamental weakness with this method is that it requires, for solutions to be unique, the energy landscape to be generally convex. Because of this constraint, it is often times not a good method to use on many problems, which have more than one comparable energy wells. Most functions that this method is used on are not well understood, and so such an observation may not be made. The method is highly dependant on the way that the vectors are mutated. It is possible for a vector which can be significantly improved by a random mutation to never undergo that mutation because the way that the mutations are done makes it impossibly unlikely for such a mutation to occur. This is known as the *encoding problem.* Finally, the final solution of a simulated annealing run is often times highly dependant on the initial vector position and the cooling schedule. This would indicate a non-convex function, and there is no way yet of overcoming this problem except to run the program from many different initial conditions.

4 Genetic Algorithms

4.1 Definition

Genetic algorithms are generalizations of natural genetic search mechanisms. In its simplest form, a genetic algorithm may be built over a finite alphabet $A = \{a_1, \ldots, a_n\}$ and using strings of length ℓ built using the alphabet A. This is a *population-based* *search algorithm*, and many of the concerns in using the genetic algorithm are those found while using general population based search algorithms. We start with a population P, made up of many strings of length ℓ and built using the alphabet A. A function known as the *fitness function* or the *energy function* maps each possible string to a real number. Our objective is to find the string which maximizes or minimizes (as the problem dictates) the fitness function.

There are three operators defined to help us. The first is the *mutation operator*. This operator randomly chooses one or more of the string positions and replaces the current letter with a new letter from the alphabet. This serves as a search mechanism which 'finds' new information in the fitness function. The second is the *crossover operator*. This operator chooses one or more of the letters in a word, and chooses two elements of the population, and switches the chosen letters in the two elements. This serves as a sorting operator, which sorts out the best information in the population and stores it in a group of elements (a subpopulation). The third is the *reproduction operator*. This serves as the 'engine' that drives the method. This removes those elements from the population that do not score as well, probabilistically, in favor of the higher scoring elements.



General Genetic Algoritm Flowchart

4.2 Strengths and Weaknesses of the Method

Of all the methods discussed so far, the genetic algorithm is the most robust. In fact, as mentioned above, it can even be shown to be equivalent to both of the methods given above. This method is capable of passing over local minima and maxima at varying degrees, depending on the way that the reproduction operator is encoded. The search ability of the genetic algorithm is dictated by two operators. The mutation operator introduces new information into the population. The crossover operator sorts through this information, producing new elements with different ways of combining this information. These two operators perform the search very robustly because, when the function is separable, that is when the function can be written as the sum of two different functions, each of which depend on different subsets of the strings, these different subsets of strings can be efficiently sorted. In this way, improvements due to mutation in one part of the vector may be added to improvements due to a different mutation in another vector, and these would then produce a superior vector. Removal of the inferior information from the population due to the reproduction operator can be either strict or loose, or vary depending on a number of different factors. The lack of rigidity of the method makes it powerful, while its fundamental operators make it effective.

The genetic algorithm, while very powerful and robust, is limited by the encoding. Once again, the concerns that cropped up when we were considering the simulated annealing appear. (This is due to the fact that the simulated annealing is, in fact, a genetic algorithm.) The effectiveness of the mutation operator can often times be orders of magnitude smaller than that of another mutation operator working on the same problem and population. Moreover, we stated that the crossover operator is good at sorting information when the problem is separable. It is not true when the problem is not separable. The combination of difficulties in both these operators can lead to premature convergence, and non-competitive results. Finding a good encoding for these operators is a subject of current research. The genetic algorithm is also computationally dense. Like other population-based techniques, the genetic algorithm requires a great deal of computation. Depending on the fitness function in question, this computation can often times be expensive. However, once this computation is done, computations such as population-based gradients, and encoding transformations may be carried out. It is beyond the scope of this synopsis to delve into these techniques. It suffices to say that the genetic algorithm's performance benefits greatly from these concerns.

It is worth noting that the genetic algorithm is also susceptible to the problem of local maxima and minima, in certain encodings. As the genetic algorithm is also a hill-climbing method in a particular view, it cannot overcome several of the same barriers that other hill-climbing methods cannot. However, it is capable of searching, very quickly, the entire hypercube encasing all of its elements. This makes its limitation to a particular hill only true if that particular hill encompasses the entire hypercube that contains the population. By making this hypercube very large, the requirement that the entire hill be large may be made, and this can often times lead to unique solutions if the 'good' solutions are spatially bounded.

5 Conclusions

Stochastic methods were developed in order to deal with the problem of 'rugged fitness landscapes'. The methods succeed where other methods fail because they are capable of avoiding many of the traps encountered by other methods. However, this does not mean that these methods are immune to these pitfalls. The directed random walk is capable of finding precise solutions to the problem, but the solution obtained is very dependent on the initial conditions of the algorithm, and on the step size. The simulated annealing is capable of walking up, down, and over peaks in order to find new peaks. This makes it less limited by locality problems. However, it is usually only robust in its solution if the function is convex, and this makes it dependant on initial conditions. Also because it uses only one solution for its search, it cannot take advantage of population-based information. Simulated annealing also requires a cooling schedule, and it is not always clear what this should entail. Finally, the genetic algorithm is quite robust, and can be made to run the same calculations as the random walk and simulated annealing searches. However, it is susceptible (to a smaller degree) to locality problems, and to encoding problems.