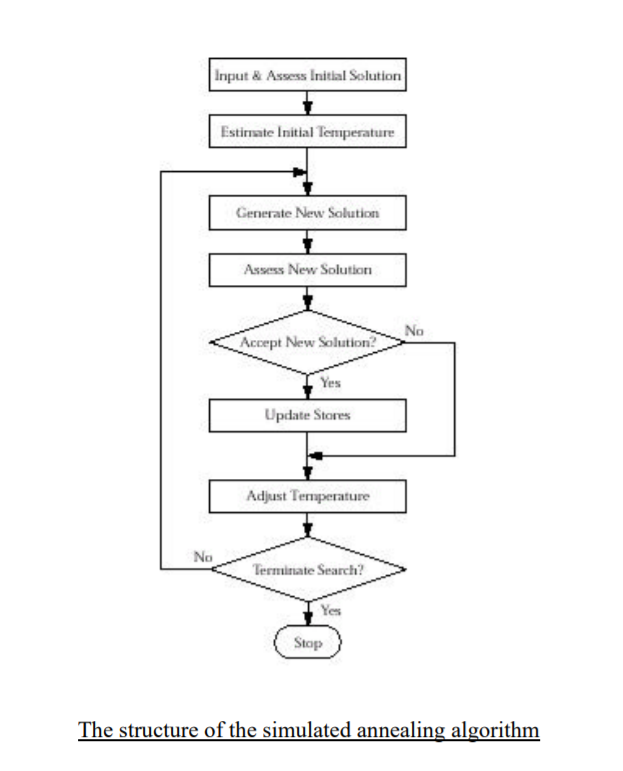
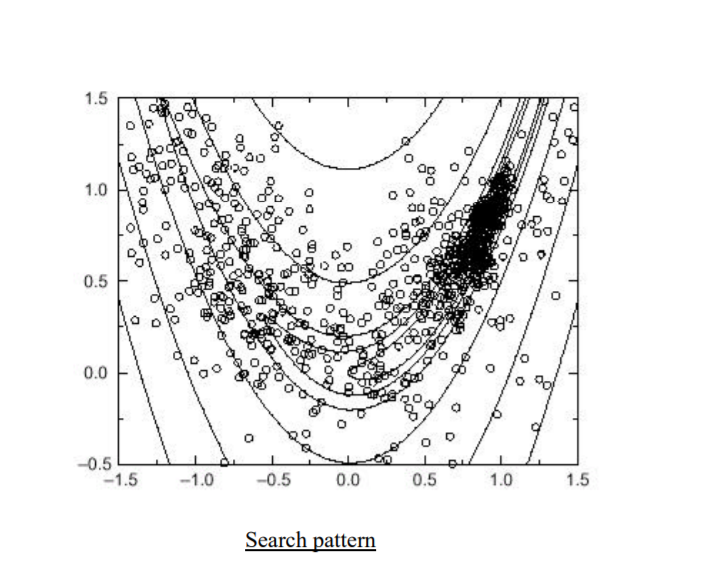
**Simulated Annealing**

Simulated annealing (SA) is a random-search technique which exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system; it forms the basis of an optimisation technique for combinatorial and other problems.

Simulated annealing was developed in 1983 to deal with highly nonlinear problems. SA approaches the global maximisation problem similarly to using a bouncing ball that can bounce over mountains from valley to valley. It begins at a high "temperature" which enables the ball to make very high bounces, which enables it to bounce over any mountain to access any valley, given enough bounces. As the temperature declines the ball cannot bounce so high, and it can also settle to become trapped in relatively small ranges of valleys. A generating distribution generates possible valleys or states to be explored. An acceptance distribution is also defined, which depends on the difference between the function value of the present generated valley to be explored and the last saved lowest valley. The acceptance distribution decides probabilistically whether to stay in a new lower valley or to bounce out of it. All the generating and acceptance distributions depend on the temperature.

It has been proved that by carefully controlling the rate of cooling of the temperature, SA can find the global optimum. However, this requires infinite time.





**Advantages**

Simulated annealing can deal with highly nonlinear models, chaotic and noisy data and many constraints. It is a robust and general technique.

Its main advantages over other local search methods are its flexibility and its ability to approach global optimality.

The algorithm is quite versatile since it does not rely on any restrictive properties of the model.

SA methods are easily "tuned". For any reasonably difficult nonlinear or stochastic system, a given optimisation algorithm can be tuned to enhance its performance and since it takes time and effort to become familiar with a given code, the ability to tune a given algorithm for use in more than one problem should be considered an important feature of an algorithm.

**Disadvantages**

Since SA is a meta-heuristic, a lot of choices are required to turn it into an actual algorithm.

There is a clear trade-off between the quality of the solutions and the time required to compute them.

The tailoring work required to account for different classes of constraints and to fine-tune the parameters of the algorithm can be rather delicate.

The precision of the numbers used in implementation is of SA can have a significant effect upon the quality of the outcome

**Comparison with other methods**

Any efficient optimisation algorithm must use two techniques to find a global maximum: exploration to investigate new and unknown areas in the search space, and exploitation to make use of knowledge found at points previously visited to help find better points. These two requirements are contradictory, and a good search algorithm must find a tradeoff between the two.

**Random Search – Hill Climbing**

The brute force approach for difficult functions is a random, or an enumerated search. Points in the search space are selected randomly, or in some systematic way, and their fitness evaluated. This is an unintelligent strategy, and is rarely used by itself.

**Genetic algorithms**

A Genetic Algorithm maintains a population of possible solutions, and at each step, selects pairs of possible solution, combines them (crossover), and applies some random changes (mutation). The algorithm is based on the idea of "survival of the fittest" where the selection process is done according to a fitness criteria (usually in optimization problems it is simply the value of the objective function evaluated using the current solution). The crossover is done in hope that two good solutions, when combined, might give even better solution.

On the other hand, Simulated Annealing only tracks one solution in the space of possible solutions, and at each iteration considers whether to move to a neighbouring solution or stay in the current one according to some probabilities (which decays over time). This is different from a heuristic search (say greedy search) in that it doesn't suffer from the problems of local optimum since it can get unstuck from cases where all neighbouring solutions are worst than the current one.

In practice: **GA nearly always beats SA** (returns a lower 'best' return value from the cost function--ie, a value close to the solution space's global minimum), but at a higher computation cost. It means GA computation can be distributed, which means in practice, we can get much better results (closer to the global minimum) and better performance (execution speed).

**In what circumstances might SA outperform GA?**

The general scenario where SA outperforms GA would be those optimization problems having a small solution space so that the result from SA and GA are practically the same, yet the execution context (e.g., hundreds of similar problems run in batch mode) favours the faster algorithm (which should always be SA).