International Journal of Advanced Research in Engineering and Applied Sciences

ISSN: 2278-6252
Impact Factor: 7.388

DERIVATIVE FREE OPTIMIZATION BY USING SIMULATED ANNEALING METHOD

FIRAOL ASFAW WODAJO - Department of Mathematics, Debre Berhan University, Ethiopia

ABSTRACT

Annealing is the physical process of heating up a solid until it melts, followed by cooling it down until it crystallizes into a state with a perfect lattice. During this process, the free energy of solid is minimized. Practice shows that the cooling must be done carefully in order not to get trapped in a locally optimal lattice structure with crystal imperfections. In combinatorial optimization, we can define a similar process. This process can be formulated as the problem of finding (among a potentially very large number of solutions) a solution with minimal cost. Now, by establishing a correspondence between the cost function and the free energy, and between the solutions and physical states, we can introduce a solution method in the field of combinatorial optimization based on a simulation of the physical annealing process. The resulting method is called Simulated Annealing (SA).

KEYWORDS: Simulation, Simulation Annealing, Boltzmann's Constant, Optimization, Dejong's fifth function

1.1. Simulated Annealing

The simulated annealing method is based on the simulation of thermal annealing of critically heated solids. When a solid (metal) is brought in to a molten state by heating it to a high temperature, the atoms in the molten metal move freely with respect to each other: However, the moots of atoms get restricted as the temperature is reduced. As the temperature reduces, the atoms tend to get ordered and finally form crystals having the minimum possible internal energy. The process of formation of crystals essentially depends on the cooling rate. When the temperature of the molten metal is reduced at a very fast rate, it may not be able to crystalline state having a higher energy state compared to that of the crystalline state.

The salient features of SA method may be summarized as follows.

- It could find a high-quality solution that does not strongly depend on the choice of the initial solution.

- It does not need a complicated mathematical model of the problem under study.
- It starts with any given solution and try to improve it. This feature could be utilized to improve a solution obtained from other suboptimal or heuristic methods.
- It has been theoretically proved to converge to the optimum solution.

1.2.1. Procedures of SA

The simulated annealing method simulated the process of slow cooling of molten metal to achieve the minimum function value in a minimization problem. The cooling phenomenon of the molten metal is simulated by using the concept of $Boltzmann's$ probability distribution. The Boltzmann's probability in distribution implies that the energy (E) of a system in thermal equilibrium at temperature T is distributed probabilistically according to the relation

$$P\ (\ E) = \ e^{-\ \frac{E}{kT}} \qquad (1.1\ )$$

Where $P(E)$ denotes the probability of achieve the energy level E, and K is called the Boltzmann's constant. equation $(\ 4.1)$ shows that at high temperature the system has nearly a uniform probability of being at a high energy state. However, at low temperatures, the system has small probability of being at a high-energy state. This indicates that when the search process is assumed to follow Boltzmann's probability distribution, the convergence of the simulated annealing algorithm can be controlled by controlling the temperature T. The method of implementing the Boltzmann's probability distribution in simulated thermodynamic systems, suggested by Metropolis et al, can also be used in the context of minimization of functions. In the case of function minimization, let the current design point (state) be x_i, with the corresponding value of the objective function given by

$$f_i \ = \ f(x_i) \qquad (1.2)$$

Similar to the energy state of a thermo dynamic system, the energy E_i at state x_i is given by

$$E_i \ = \ f_i = \ f(x_i) \qquad (1.3)$$

Then according to the Metropolis criterion, the probability of the next design point (state) x_{i+1} depends on the difference in the energy state or function values at the two design points (states) given by:

$$\Delta E = E_{i+1} - E_i \ = \Delta f = f_{i+1} - f_i \equiv f(x_{i+1}) - f(x_i) \qquad (1.4)$$

The new state design point x_{i+1} can be found using Boltzmann's probability distribution:

$$P[E_{i+1}]=min\left\{1,e^{\frac{-\Delta E}{KT}}\right\} \quad (1.5)$$

The Boltzmann's constant serves as a scaling factor in simulated annealing and as such, can be chosen as 1 for simplicity. Note that if $\Delta E \leq 0 \text{equation}(4.5)$ gives $P[E_{i+1}] = 1$ and hence the point x_{i+1} is always accepted. This is a logical choice in the context of minimization of a function because the function value at x_{i+1} ,f_{i+1}, is better(smaller) than at x_i.f_i, and hence the design vector x_{i+1}must be accepted. On the other hand, when$\Delta E > 0$, the function value f_{i+1}at x_{i+1} is worse(larger) than the one at x_i. According to most conventional optimization procedures, the point x_{i+1} cannot be accepted as the next point in the iterative process. However, the probability of accepting the point x_{i+1} , inspite of its being worse than x_i in terms of the objective function value,is finite (although it may be small) according to the Metropolis criterion. Note that the probability of accepting the point x_{i+1}.

$$P[E_{i+1}] = min\left\{e^{\frac{-\Delta E}{KT}}\right\} \quad (1.6)$$

is not the same in all situations. As can be seen from $\text{equation}(4.6)$ this probability depends on the values ΔE and T. If the temperature T is large, the probability will be high for design points x_{i+1} with larger function values (with larger values of $(\Delta E = \Delta f)$.Thus, at high temperatures, even worse design points x_{i+1} (with large values of $(\Delta E = \Delta f)$ will be small. Thus, as the temperature values get smaller (that is, as the process gets closer to the optimum solution), the design points x_{i+1} with larger function values compared to the one at x_i are less likely to be accepted.

1.2.2. Algorithm

The SA algorithm can be summarized as follows. Start with an initial design vector x_1 (iteration number $i = 1$) and a high value of temperature T. Generate a new design point randomly in the vicinity of the current design point and find the difference in function values;

$$\Delta E = \Delta f = f_{i+1} - f_i \equiv f(x_{i+1}) - f(x_i) \quad (1.7)$$

If f_{i+1} is smaller than f_i (with a negative value of Δf, accept the point x_{i+1} as the next design point. Otherwise, when Δf is positive, accept the point x_{i+1}as the next design point

only with a probability $e^{-\frac{\Delta E}{KT}}$. This means, that if the value of a randomly generated number is larger than $e^{-\frac{\Delta E}{KT}}$, accept the point x_{i+1}. Otherwise, reject the point x_{i+1}. This completes by one iteration of SA algorithm. If the point x_{i+1} is rejected, then the process of generating a new design point x_{i+1} randomly in the vicinity of the current design point, evaluating the corresponding objective function value f_{i+1},and deciding to accept x_{i+1} as the new design points based on the use of the metropolis criterion ,equation(4.6) is continued. To simulate the attainment of thermal equilibrium at every temperature, a predetermined number (n) of new points x_{i+1} are tested at any specific value of the temperature$T$. Once the number of new design points x_{i+1} tested at any temperature T exceeds the value of n, the temperature T is reduced by a pre specified fractional value c $(0 < c < 1)$ and the whole process is repeated. The procedure is assumed to have converged when the current value of temperature T is sufficiently small or when changes in the function values (Δf) are observed to be sufficiently small. The choices of the initial temperature T,the number of iterations n before reducing the temperature, and the temperature reduction factors c play important roles in the successful of the SA algorithm. For example, if the initial temperature T is too large, it requires a large number of temperature reductions for convergence. On the other hand, if the initial temperature is chosen to be too small, the search process may be incomplete in the sense that it might fail to thoroughly investigate the design space in locating the global minimum before convergence. The temperature reduction factor c has a similar effect. Too large a value of c (such as 0.8 or 0.9) requires too much computational effort for convergence. On the other hand, too small a value of c(such as 0.1 or 0.2) may result in a faster reduction in temperature that might not permit a thorough exploration of the design space for locating the global minimum solution. Similarly, a large value of the number of iterations n will help in achieving quasi equilibrium state at each temperature level but will result in a larger computational effort. A smaller value of n , on the other hand, might result either in a premature convergence or convergence to a local minimum (due to inadequate exploration of the design space for the global minimum).Unfortunately, no unique set of values are available for T,n and c that will work well for every problem. However, certain guidelines can be given for selecting these values. The initial temperature T can be chosen as the average value of the objective function computed at a number of randomly selected points in the design space. The number of iterations n can be

chosen between 50 and 100 based on the computing resources and the desired accuracy of solution. The temperature reduction factor c can be chosen between 0.4 and 0.6 for a reasonable temperature reduction strategy (also termed the cooling schedule). More complex cooling schedules, based on the expected mathematical convergence rates, have been used in the literature for the solution of complex practical optimization problems. In spite of all the research being done on SA algorithms, the choice of the initial temperature T, the number of iterations n at any specific temperature ,and the temperature reduction factor(or cooling rate) c still remain an art and generally require a trial and error process to find suitable values for solving any particular type of optimization problems.

1.2.3. Features of the method

Some of the features of simulated annealing are as follows.

1. The quality of the final solution is not affected by the initial guesses, except that the computational effort may increase with worse starting designs.

2. Because of the discrete nature of the function and constraint evaluations, the convergence or transition characteristics are not affected by the continuity and differentiability of the functions.

3. The convergence is also not influenced by the convexity status of the feasible space.

4. The design variables need not be positive.

5. The method can be used to solve mixed integer, discrete, or continuous problems.

6. For problems involving behavior constraints (in addition to lower and upper bounds on th e design variables), an equivalent unconstrained function is to be formulated as in the case of genetic algorithms.

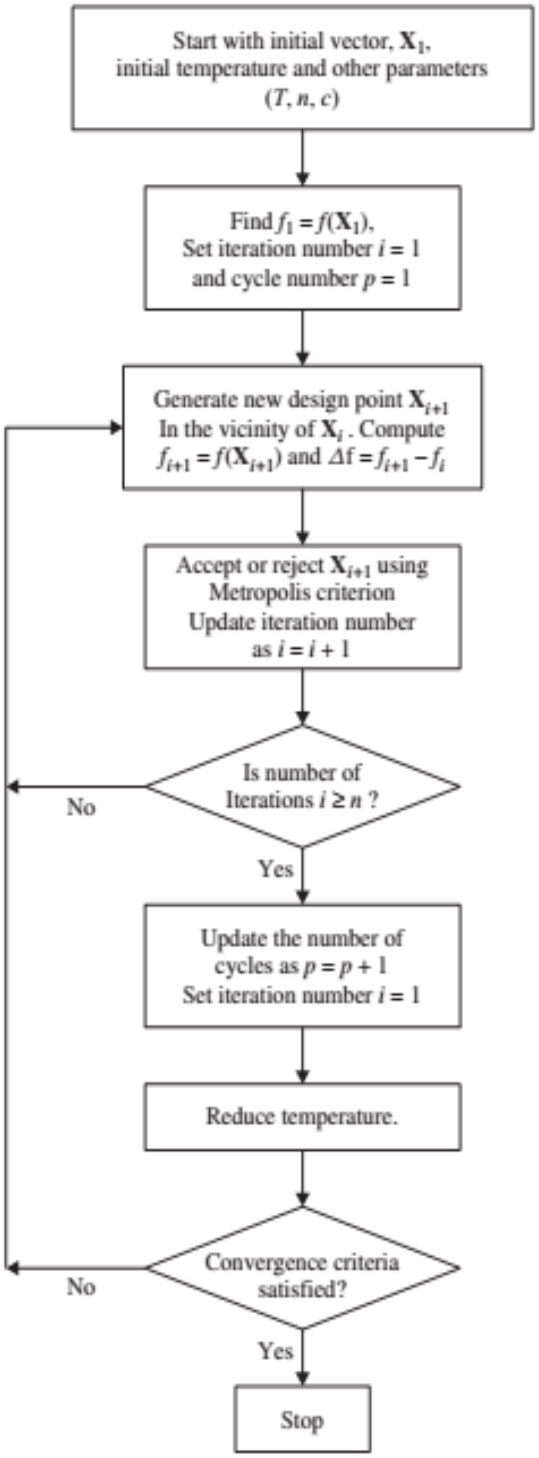


Figure 1. Simulated Annealing Procedure

1) **12.4. Some examples and applications of SA**

Simulated Annealing helps us to solve different optimization problems. Here, we try to solve few problems as follows.

Example 1

Find the minimum of the following function using simulated annealing:

$$f(x) = 500 - 20x_1 - 26x_2 - 4x_1x_2 + 4x_1^2 + 3x_2^2$$

Solution:

We follow the procedure indicated on the flow chart of fig 13.2

Step1: choose the parameters of the SA method. The initial temperature is taken as the average value of f evaluated at four randomly selected points in the design space. By selecting the random points as $x^{(1)} = \binom{2}{0}, x^{(2)} = \binom{5}{10}, x^{(3)} = \binom{8}{5}, x^{(4)} = \binom{10}{10}$, we find the corresponding values of the objective function as

$f^{(1)} = 476, \quad f^{(2)} = 340, \quad f^{(3)} = 381, \quad f^{(4)} = 340,$

respectively. Noting that the average value of the objective functions $f^{(1)}, f^{(2)}, f^{(3)}$ and $f^{(4)}$ is 384.25, we assume the initial temperature to be $T = 384.25$. The temperature reduction factor is chosen as c=0.5.To make the computations brief, we choose the maximum permissible number of iterations (at any specific value of temperature) as n=2.)

Step 2: Evaluate the objective function value at x_i as

$f_1 = 349.0$ and set the iteration number is $i = 1$

Step 3: Generate a new design point in the vicinity of the current design point. For this, we select two uniformly distributed random numbers u_1 and u_2 for x_1 in the vicinity of 4 and u_2 for x_2 in the vicinity of 5.The numbers u_1 and u_2 are 0.31 and 0.57, respectively. By choosing the ranges of x_1 and x_2 as (-2, 10) and (-1, 11),which represent ranges of $\pm$ 6 about their respective current values, the uniformly distributed random numbers r_1 and r_2 in the ranges of x and x, corresponding to u_1 and u_2 ,can be found as

$r_1 = -2 + u_1\{10 - (-2)\} = -2 + 0.31(12) = 1.72$,

$r_2 = -1 + u_2\{11 - (-1)\} = -1 + 0.57(12) = 5.84$ which gives $x_2 = \binom{r_1}{r_2} = \binom{1.72}{5.84}$

Since the objective function value $f_2 = f(x_2) =$ 387.7312, the value of Δf is given by

$$\Delta f = f_1 - f_2 = 387.7312 - 349.0 = 38.731$$

Step 4: Since the value of Δf is Positive, we use the Metropolis criterion to decide whether to accept or reject the current point. For this we choose a random number in the range (0,1) as $r = 0.83$ .Equation (4.6) gives the probability of accepting the new design point x_2 as

$$P[x_2] = e^{\frac{-\Delta f}{KT}} \qquad (1.8)$$

By assuming the value of the Boltzmann's constant k to be 1 for simplicity in equation(4.8, we obtain

$$P[x_2] = e^{\frac{-\Delta f}{KT}} = e^{-38.7312/384.25} = 0.904$$

Since $r = 0.8$ is smaller than 0.9041, we accept the point $x_2 = \binom{1.72}{5.84}$ as the next design p oint. Note that, although the objective function value f_2 is larger than f_1, we accept x_2. because this is an early stage of simulation and the current temperature is high.

Step 3: Update the iteration number as $i = 2$
Since the iteration number is i $\leq$ n , we proceed to step - 3
$Step\ 3$: Generate a new design point in the vicinity of the current design point $x_2 = \binom{1.72}{5.84}$. For this, we choose the range of each design variable as ± 6 about its current value so that the ranges are given by (-6+1.72 ,6+1.72)=(-4.28,7.72) for x_1 and (-6+5.84 , 6+5.84)=(-0.16 , 11.84) for x_2
.By two uniformly distributed random numbers in the range (0, 1) as

$u_1 = 0.92$ and $u_2 = 0.73$

The corresponding uniformly distributed random numbers in the ranges of x_1 and x_2 become

$r_1 = -4.28 + u_1\{7.72 - (-4.28)\} = -4.28 + 0.92(12) = 6.76$
$r_2 = -0.16 + u_2\{11.84 - (-0.16)\} = -0.16 + 0.73(12) = 8.6$

Which gives:
$x_3 = \binom{r_1}{r_2} = \binom{6.76}{8.60}$ with a function value of $f_3 = 313.3264$ .We note that the function value f_3 is better than f_2 with $\Delta f = f_3 - f_2 = 313.3264 - 387.7312 = -74.4048$
Step 4: Since $\Delta f < 0$, we accept the current point as
$x_3$and increase the iteration number to i=3, since i> n, we go to step 5.
Step 5: Since a cycle of iterations with the current value of temperature is completed, we re duce the temperature to a new value of $T = 0.5(384.25) = 192.125$. Reset the current it eration number as i=1 and go to *step 3.*

Step 3: Generate a new design point in the vicinity of the current design point x_i and contin ue the procedure until the temperature is reduced to a small value (until convergence).

Example 2.

This example shows how to find a local minimum of a function using Simulated Annealing.

Dejong's fifth function is a two-dimensional function with many (25) local minima:

dejong5fcn

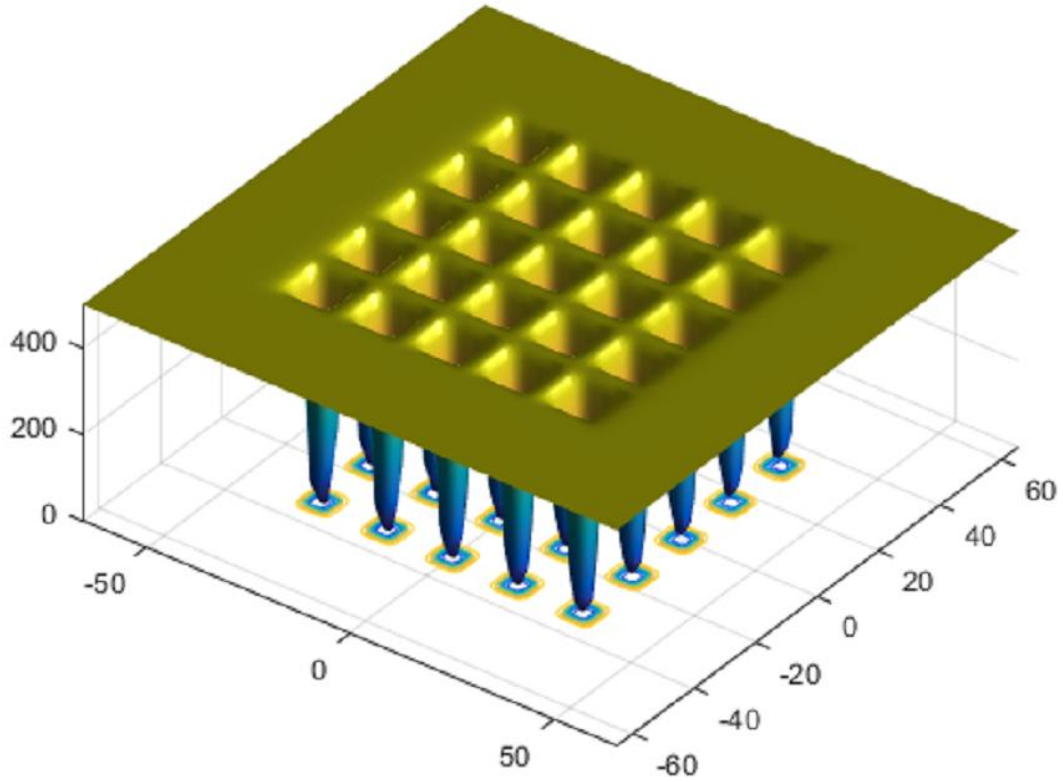


Figure 2. Graph of dejong's fifth function

Many standard optimization algorithms get stuck in a local minima. Because the simulated annealing algorithm performs a wide random search, the chance of being trapped in local minima is decreased.

Note: Because Simulated Annealing uses random number of generators, each time you run this algorithm you can get different results.

Minimize at the Command Line

To run the simulated annealing algorithm without constraints, call **simulannealbnd**at the command line using the objective function in dejong5fcn.m,referenced by anonymous function pointer:

```
rng(10,'twister') % for reproducibility
fun = @dejong5fcn;
[x fval] = simulannealbnd (fun,[0 0])
```

This returns

```
Optimization terminated: change in best function value less than options. Tol Fun.
x =  -16.1292  -15.8214
fval =  6.9034
```

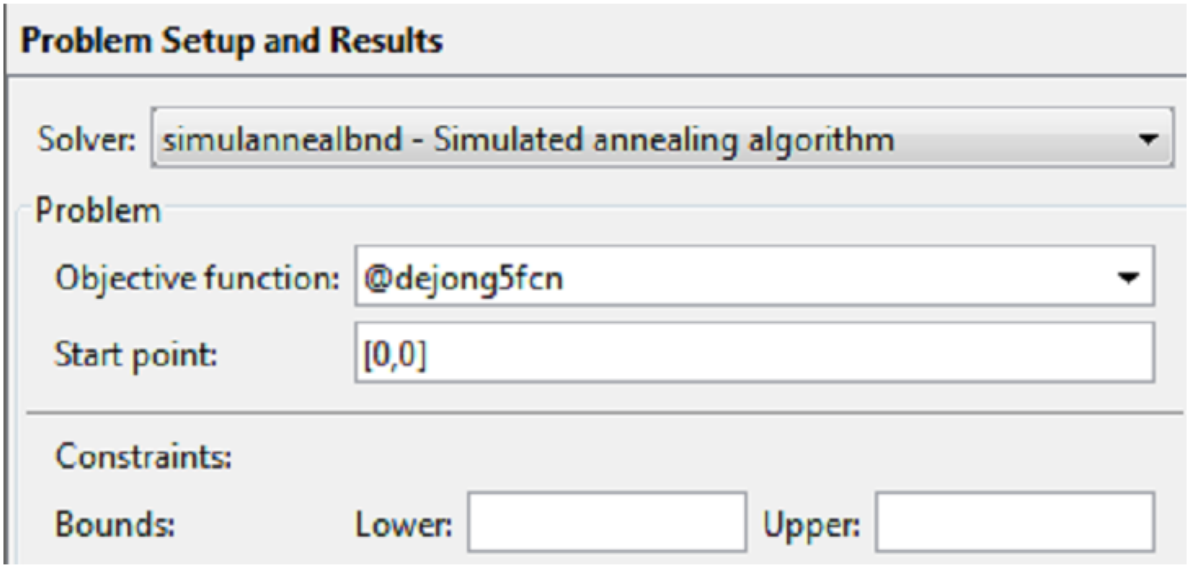
Where:

- x is the final point returned by the algorithm.
- fval is the objective function value at the final point.

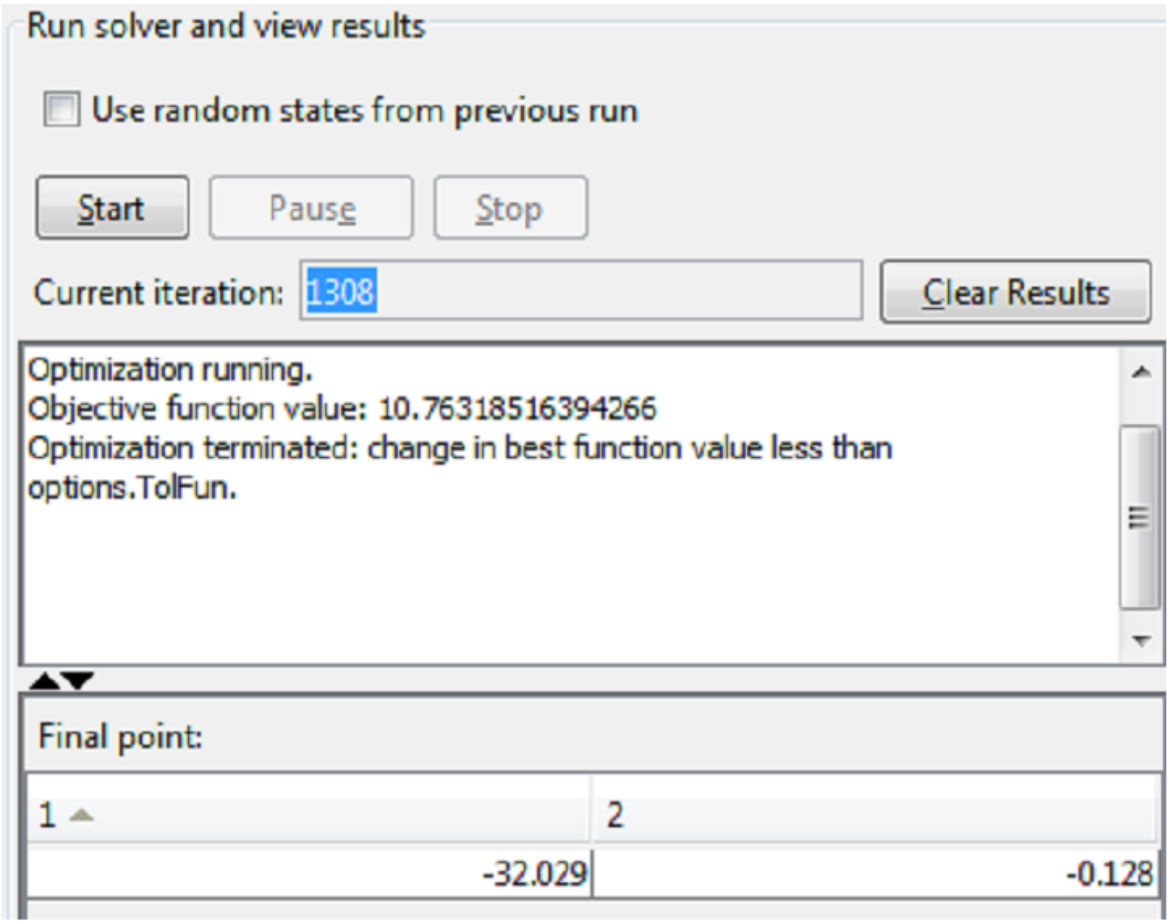
Minimize Using Optimization Application

To run the minimization using the Optimization application

1. Set up your problem as pictured in the Optimization application



2. Click **Start** under **Run solver and view results**:



Your results can differ from the pictured ones, because **simulannealbnd** uses a random number stream.

Example 3.

Hybridization of Simulated Annealing and fminunc

This example uses Optimizaton Toolbox function fminunc, an unconstrained minimization function. The example first runs the genetic algorithm to find a point close to the optimal point and then uses that point as the initial point for fminunc. The example finds the minimum of Rosebrock's function is defined by

$f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (x_1)^2.$

The following shows the plot of Rosebrock's function

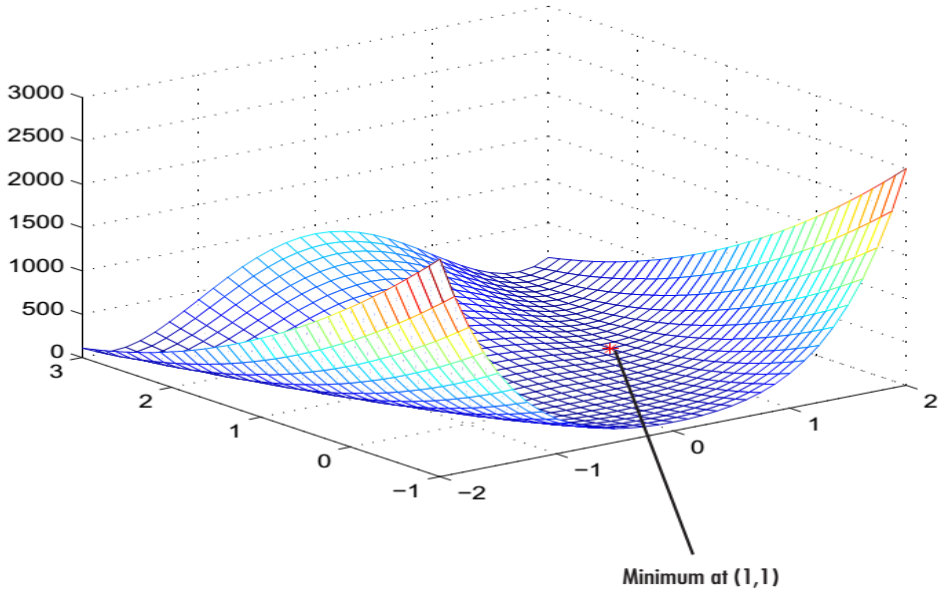


Figure 3.Graph of of Rosebrock's function

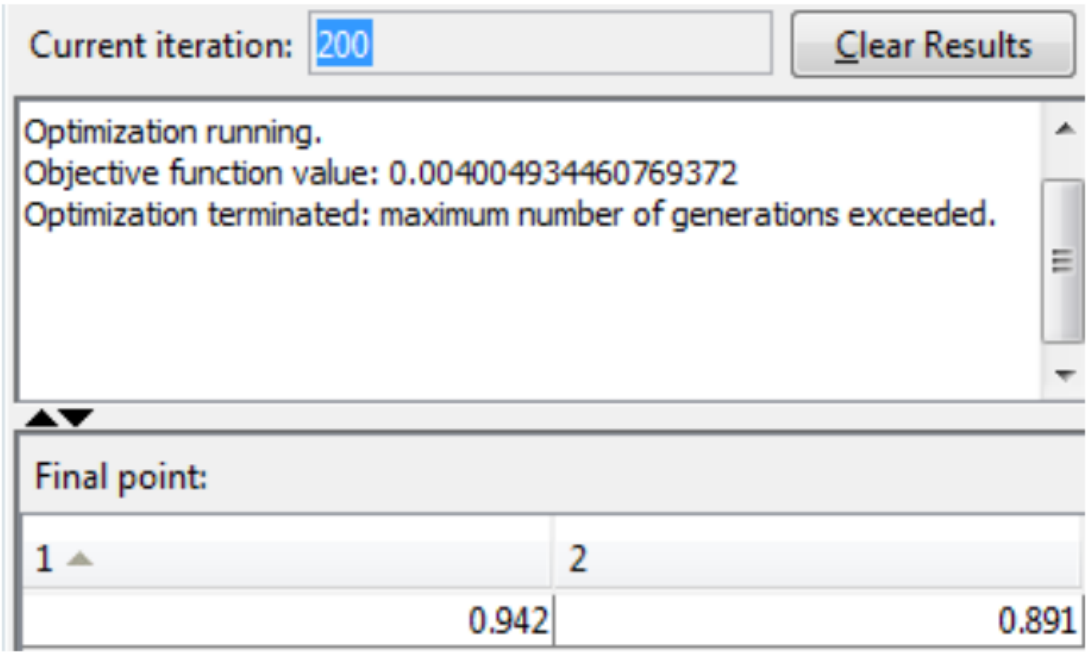
Global Optimization Toolbox software the dejong 2 fcn.m file, which computes Rosebrock's function. To explore the example, first enter optimtool('ga') to open the Optimization app to the ga solver. Enter the following settings:

- Set **Fitness function** to @dejong2fcn.
- Set **Number of variables** to 2.
- Optionally, to get the same pseudorandom numbers as this example, switch to the command line and enter:

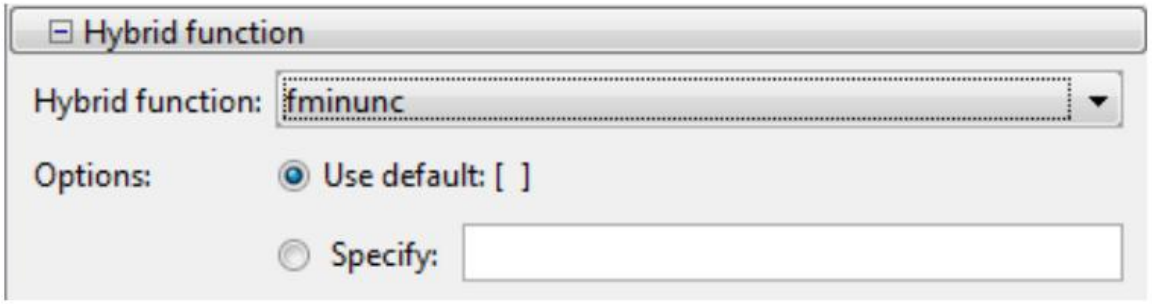
  rng(1, 'twister')

Before adding a hybrid function, click **Start** to run the genetic algorithm by itself. The genetic algorithm displays the following results in the **Run solver and view results** pane:

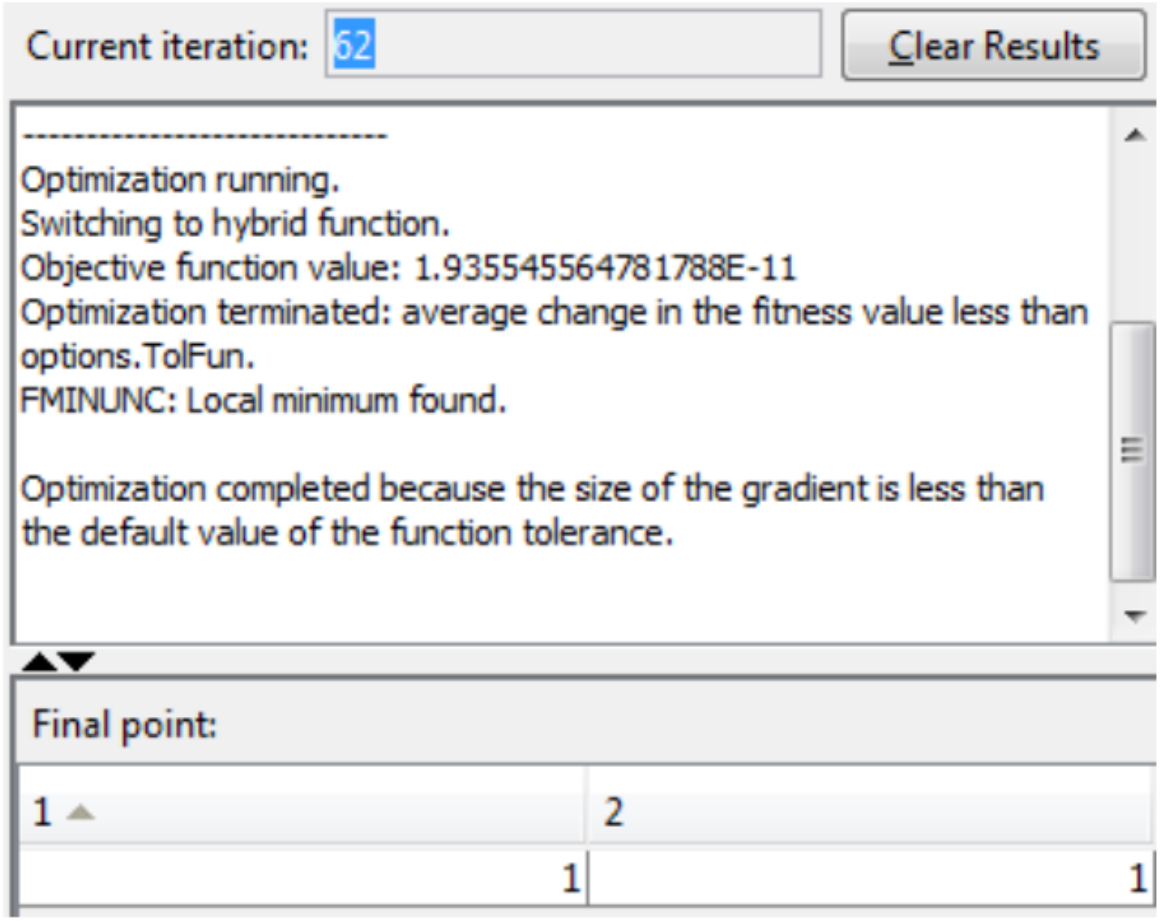
ISSN: 2278-6252
Impact Factor: 7.388



The final point is somewhat close to the true minimum at (1,1). You can improve this result by setting **Hybrid function** to fminunc(in the **Hybrid function** options).



fminunc uses the final point of the genetic algorithm as its initial point. I t returns a more accurate result, as shown in the **Run solver and view results** pane.



Example 4

Minimization Using simulated Annealing Algorithm

This example shows how to create and minimize an objective function using Simulated Annealing in the Global Optimization Toolbox

A simple Objective Function

We want to minimize a simple function of two variables

min f(x) = (4 - 2.1*x1^2 + x1^4/3)*x1^2 + x1*x2 + (-4 + 4*x2^2)*x2^2;

Coding the objective Function

We create MATLAB file named simple objective. M with the following code in it:

Function simple_ objective (x)

y = (4 - 2.1*x(1)^2 + x(1)^4/3)*x(1)^2 + x(1)*x(2) + ...(-4 + 4*x(2)^2)*x(2)^2;

The Simulated Annealing solver assumes the objective function will take one input x where x has as many elements as the number of variables in the problem. The objective function computes the scalar value of the objective and returns it in its single return argument y.

Minimizing Using SIMULANNEALBND

To minimize our objective function using the SIMULANNEALBND function, we need to pass in a function handle to the objective function as well as specifying a start point as the second argument.

```
ObjectiveFunction=@simple_objective;
X0=[0.5,0.5];%Starting point
[x,fval,exitFlag,output] =  simulannealbnd(ObjectiveFunction,X0)
Optimization terminated: change in best function value less than options.TolFun.
x = -0.0896    0.7130
fval =  -1.0316
exitFlag =   1
output =
iterations: 2948
funccount: 2971
message: [1x80 char]
rngstate: [1x1 struct]
problemtype: 'unconstrained'
temperature: [2x1 double]
totaltime: 2.1094
```

The first two output arguments returned by SIMULANNEALBND are x, the best point found, andfval, the function value at the best point. A third output argument, exit Flagreturn a flag corresponding to the reason SIMULANNEALBND stopped. SIMULANNEALBND can also return a fourth argument, output, which contains information about the performance of the solver.

SUMMARY

Simulated annealing method is based on the simulation of thermal annealing of critically heated atoms in the molten metal move freely with respect to each other.The simulated annealing method simulated the process of slow cooling of molten metal to achieve the minimum function value in a minimization problem. The cooling phenomenon of the molten metal is simulated by using the concept of $Boltzmann's$ probability distribution. The Boltzmann's probability in distribution implies that the energy (E) of a system in thermal equilibrium at temperature T is distributed probabilistically according to the relation $P\ (\ E) = e^{\frac{-E}{KT}}$where $P(E)$ denotes the probability of achieve the energy level E, and K is called the Boltzmann's constant. This formula shows that at high temperature the system has nearly a uniform probability of being at a high energy state. However, at low

temperatures, the system has small probability of being at a high-energy state. This indicates that when the search process is assumed to follow Boltzmann's probability distribution, the convergence of the simulated annealing algorithm can be controlled by controlling the temperature T. The method of implementing the Boltzmann's probability distribution in simulated thermodynamic systems, suggested by Metropolis et al, can also be used in the context of minimization of functions.

REFERENCES

1. Nelder JA and Mead R.(1965). *A simplex method for function minimization*. Computer Journal 7: 308–313
2. Shoemaker C.A. 2005. Constrained Global Optimization of Expensive Black Functions Using Radical Basis Functions. *Journal of Global Optimization*, 31:153-171
3. Storn R and Price K (1997). *Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J.* Global Optimization 11: 341–359
4. Van Laarhoven P.and Aarts, E. 1987. *Simulated Annealing: Theory and Applications*, D. Reidel, Boston.
5. Wilson E.O. 1975). *Sociobiology:* The New Synthesis. Belknap Press, Cambridge, MA.
6. Zill, D.G.1992.*Advanced Engineering Mathematics*. PWS, Boston.