AER 501 Assignment #3

David R. Islip

December 2017

1 Objectives

The objective of this report is to showcase a standard implementation of a simulated annealing algorithm and then apply it to the minimization of the two-dimensional bump function. Furthermore, the simulated annealing algorithm is used in conjunction with a finite element model to minimize the weight of a truss structure subjected to loading such that the material does not yield.

2 Code Description

As suggested by the assignment outline, the MATLAB code is broken up into modular sections. These sections are as follows:

- SA.m (simulated annealing algorithm)
- bump.m (bump-objective function)
- trussweight.m (computation of truss weight)
- schedule.m (Temperature schedule for simulated annealing)
- move.m (generate a permissible random perturbation to a decision vector)
- main.m (perform optimizations of the objective functions as well as statistical tests on specific parameter values)

There are other functions used in the code however they are either for the purpose of generating plots or they have been discussed in previous assignments. The functional dependencies between the code sections is depicted in figure 1:

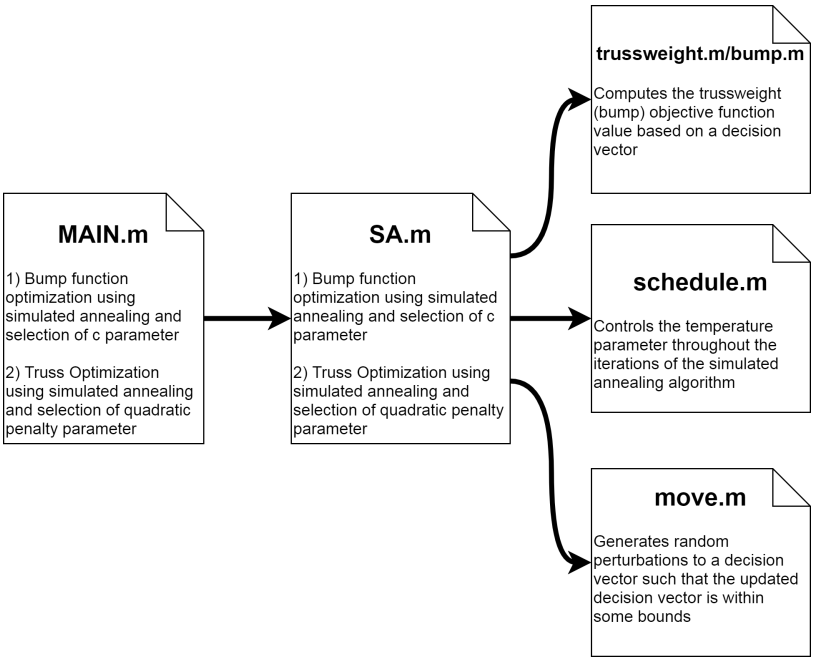


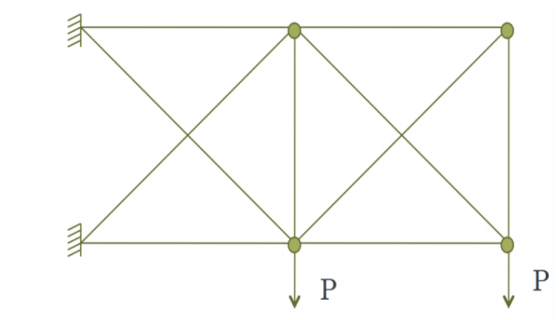
Figure 1:

2.1 Function Descriptions

2.1.1 main.m

This script minimizes the bump function subject to the constraints outlined in the assignment guideline. Numerical experiments are then performed to search for "good" value for the cooling schedule parameter. To perform the numerical experiments; simulated annealing (SA) was used N times for $c = (0.80, 0.99, 0.999)$. Based on the N objective function values obtained $3 == len(c)$ histograms are plotted to see which parameter has the highest probability of achieving the lowest objective function value.

After conducting the numerical experiments for the c parameter. SA is then applied to minimize the weight of a ten bar aluminium truss with two point loads. The truss specification is given in figure 2. SA is then used to determine the cross-section area of each element such that the tensile or compressive stress does not exceed the yield stress. Similar to the bump function minimization, SA is used repeatedly to compare the optimization results for different constraint penalty objective functions. The direct penalty formulation with a penalty magnitude of $M = 10^8$ and four other quadratic penalty formulations with penalty parameters given by $P = (1, 100, 10000, 10^6)$ are compared. Based on the computing N objective function values via SA; $1 + len(P)$ histograms are plotted to see which parameter has the highest probability of achieving the lowest objective function value.



Yield Stress: 276MPa
Young's Modulus: 70GPa
P: 100N
Buckling: Not Considered
Density: 2,700 kg/m3

Figure 2:

2.1.2 [best, xbest, x, path] = SA(x0, xu, xl, epsilon, maxiter, Tstart, c, objfunct)

The SA function takes in an initial guess, upper and lower bounds, an exploratory parameter for generating moves, a max number of iterations, a starting "temperature" a cooling rate and an objective function. The SA algorithm is outlined in figure 2.1.2. The function as implemented returns:

- best - the series formed by infimum of objective function values evaluated at prior iterations
- xbest - the series of decision variables associated with each series element of "best"
- x - series formed by the decision vectors for each iteration
- path - series formed by the objective function value at each iteration

```
for i = [1,...,maxiter]:
    1. x1 = x + random perturbation (generate move)
    2. Compute f(x)
    3. dE = f(x) - f(x1)
    4. If dE > 0 then x <-- x1(accept move)
    5. Otherwise x <-- x1 with probability exp (dE/T) (probabilistic acceptance of move)
    6. T = schedule(T, t) (update temperature)
    7. data(i) = (x,f(x)) #store the path
```

2.1.3 [f] = bump(x)

The bump function refers to the two dimensional case of the bump function as specified in the assignment. The function value is modified to account for the boundary conditions via the direct penalty method.

2.1.4 [weight isvalid] = trussweight(A0)

The trussweight function takes an area vector (one entry per element) and then computes the weight of the truss as specified in figure 2. The value of the objective function is penalized based on stress constraints where the magnitude of the stress in each element much be less than the yield stress. The stresses in each element are computed based on the code in assignment 1. NOTE: THIS PENALIZED OBJECTIVE FUNCTION DOES NOT CONSIDER BUCKLING. Furthermore, this function computes a boolean variable isvalid that determines if the solution meets the stress constraints. This is used in the statistical analysis of the quadratic penalty variable

2.1.5 [T] = schedule(Tstart,c,t)

The scheduling function computes the next iteration temperature based on $T_{i+1} = cT_i$

2.1.6 [z] = move(x, lb, ub, epsilon)

The move function perturbs a random element of an input decision vector such that the resulting vector is between some lower and upper bound. The magnitude of the perturbation is controlled by epsilon.

3 Results

3.1 Bump Function

A contour plot and a heat map of the negative bump function are provided in figure 3. The heat map shows the boundary as those areas not meeting the constraints are set to 0 for simplicity of viewing. One can see that there are many local maxima and that the maximum value of the negative bump function seems to be located on the edge of a constraint boundary. The SA function was used to solve the minimization of the bump function. After repeated trials the decision vector was determined to be $x = (1.5101, 0.4978)$ with an objective function value of 0.354. Figure 4 shows the objective function value as a function of the iteration number as well as the infimum of the observed function values. Note that epsilon was set to 0.4 and the cooling parameter was set to 0.99 (justification below)

In an attempt to determine the optimal value for the cooling parameter, $N = 100$ SA runs were done for $c = (0.8, 0.99, 0.999)$. The resulting histograms of "min" objective function values for each parameter are

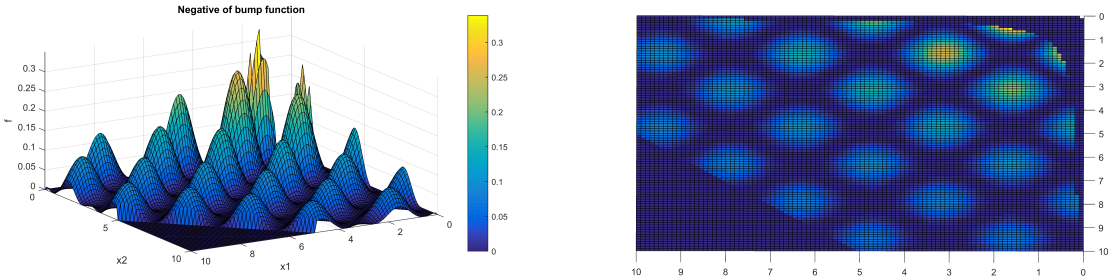


Figure 3:

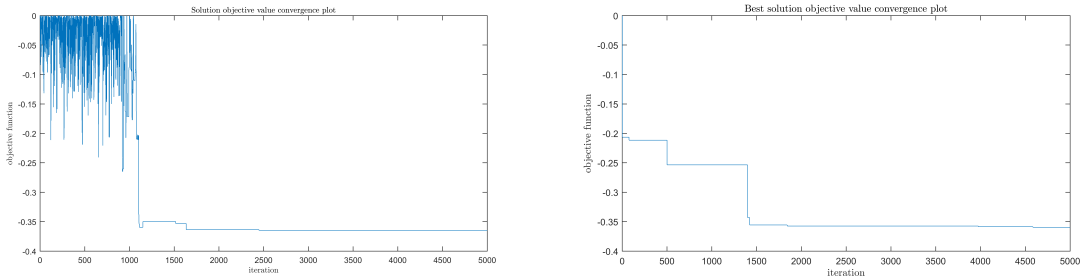


Figure 4:

shown in figure 5. It is seen that setting the cooling parameter to 0.99 will result in a minimization that is "probably" closest to the minimum objective function value. Setting the cooling parameter to 0.8, implies the probability of observing an objective function value in the lowest bucket is similar to setting the cooling parameter to 0.99; however, setting the cooling parameter to 0.8 also leads to fatter tails in the distribution of outcomes in that there is a high probability that the objective function value obtained by SA is much higher than the true minimum. Furthermore, table 1 highlights that setting the cooling parameter to 0.999 actually has the lowest median and mean objective function value of the 100 SA trials. This makes sense when one looks at the distribution in figure 5 as it seems the distribution for $c = 0.999$ is less skewed.

Table 1: averaged objective function values for various values of the cooling parameter

c	**mean objective function value**	**median objective function value**
0.8	-0.2905	-0.2699
0.99	-0.3095	-0.3075
0.999	-0.3099	-0.3149

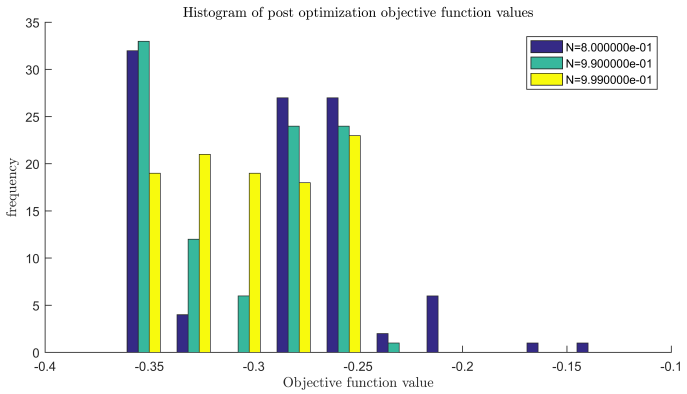


Figure 5:

3.2 Truss Weight

The SA function was used to solve the minimization of the truss structures weight. After repeated trials, the cross-sectional area was determined to be those in table 2 for various parameter selections. Figure 4 shows the truss-weight as a function of the iteration number as well as the infimum of the observed function values. Note that epsilon was set to 0.4; the cooling parameter was set to 0.99 as in the bump function case and the direct penalty method was used. Figure 7 shows the resulting optimal truss from the methodology described above. The line weights are proportional to the element area and the numbers correspond to element numbers.

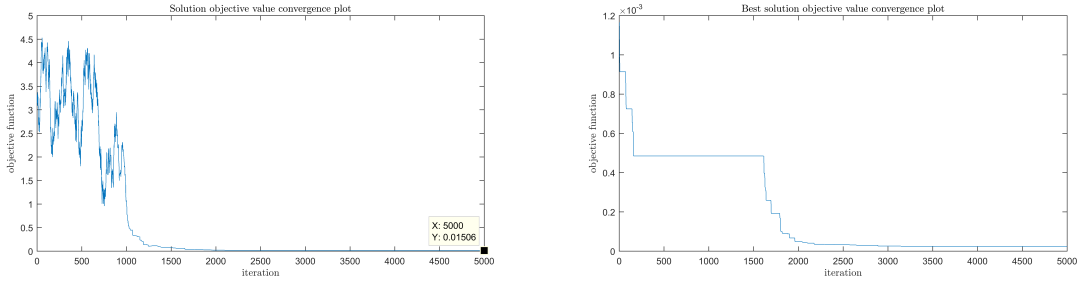


Figure 6:

Similar to the cooling parameter, a study was done in an attempt to compare the direct penalty approach against the quadratic penalty approach as well as the determine the optimal parameter value for the quadratic penalty approach. For the direct penalty approach and four instances of the quadratic approach, $N = 100$ SA runs were completed. If the solution returned by the simulated annealing algorithm violated the stress constraint it was not considered. Figure 8 highlights the results. It was observed that setting the magnitude

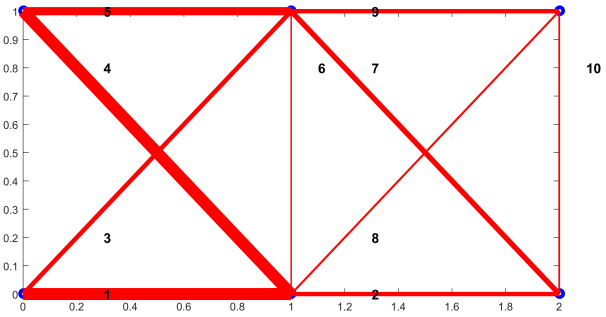


Figure 7:

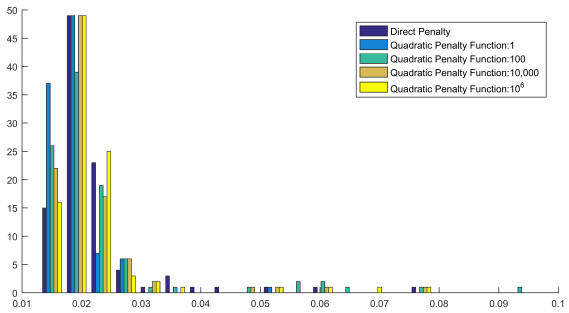


Figure 8:

of the quadratic constraint violation penalty to one obtained the highest number of simulations within the lowest bucket. Furthermore, setting the quadratic penalty parameter to one resulted in the lowest objective function value. In hindsight, I believe it would have been more useful to multiply the quadratic penalty parameter by the magnitude of the non-penalized objective function, to allow for a better sense of scale.

Table 2: Optimal solutions for each penalty parameter

	Direct Penalty	**Quadratic Penalty P = 1**	**Quadratic Penalty P = 100**	**Quadratic Penalty P = 10000**	**Quadratic Penalty P = 10^6**
Area (m2)*1.0e-05	0.0554	0.1046	0.0861	0.0972	0.0675
	0.0604	0.0669	0.0831	0.0734	0.0993
	0.1107	0.0104	0.0553	0.0283	0.1001
	0.0268	0.1151	0.0712	0.0975	0.0273
	0.1334	0.0554	0.0729	0.0624	0.0904
	0.0363	0.0255	0.0367	0.0300	0.0479
	0.0457	0.0364	0.0418	0.0385	0.0471
	0.0167	0.0268	0.0164	0.0226	0.0059
	0.0197	0.0161	0.0110	0.0141	0.0059
	0.0128	0.0471	0.0438	0.0458	0.0405
mass (kg)	0.0141	0.0130	0.0137	0.0132	0.0141