Assignment 2: CS 7641

Gandharv Kashinath

March 15, 2014

Introduction

In this assignment four randomized optimization algorithms were analyzed and evaluate for performance on three very different and interesting problems. Further, these algorithms were used to compute the weights of a neural network which was used to classify the car data set from assignment 1. The goal of this assignment was to understand the application scenarios for these optimization algorithms while understanding their underlying strengths and limitations.

This paper first gives a detailed overview of the three problems used to analyze the four optimization algorithms. This is followed by a description of the tools used to construct and analyze these algorithms. Finally, the results from the analysis are discussed and some key conclusions and insights are presented.

Function Description

Rastrigin's Function:

The Rastrigin's function is commonly used to test genetic algorithms {1}. In this assignment, a variation of the Rastrigin's function given by the following equation was used;

$$Ras(x) = 20 + 3 * (x_1^2 + x_2^2) - 10(\cos(2\pi x_1) + \cos(2\pi x_2))$$

Note that the above function is continuous but by representing floats as 32-bit vectors it can be transformed in to a dicrete-valued problem. This representation is important so that transformations like cross-over and mutations can be applied.

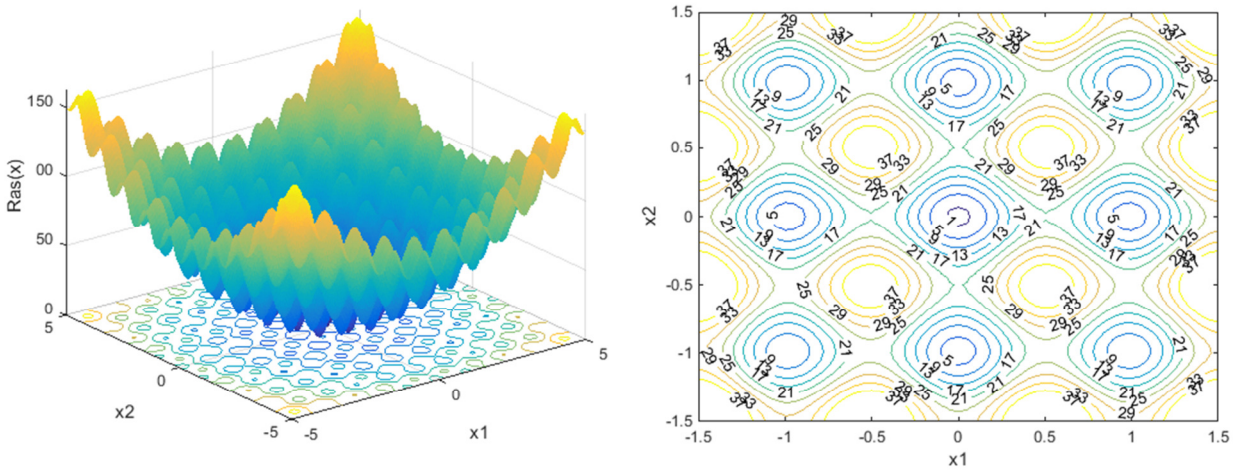

Figure 1: Rastragin's function distribution and contour showing global minimum at [0, 0].

As shown in the above figure, Rastragin's function has several local minima and one global minimum which occurs at (0, 0). At any local minima other than (0, 0), the value of Rastragin's function is greater

than 0. The farther the local minima are from the origin, the larger the value of the function is at that point.

Due to the presence of several local minima, this function is expected to do poorly for problems which rely on gradient descent methods and start from a single point. Algorithms that do well with this function are expected to do well on real-world functions that have similar structures of several local minima with one global minimum. Some real world applications include optimizing the aircraft wing airfoil profile by minimizing aerodynamic drag. Finally, although this is a minimization problem, it can be easily transformed to a maximization problem by negating the cost function.

Travelling Sales Problem:

The traveling salesman problem (TSP) consists of a salesman and a set of cities. The salesman has to visit each one of the cities (without repeating) starting from a certain one and returning to the same city {2}. The optimization goal of this problem is to minimize the total length of the trip. Although this problem is NP-complete, it is NP hard and hence not trivial to find a good solution. This problem has some really interesting applications besides the obvious salesman/travel logistics for companies. It is widely applied in identifying the optimal wiring configurations in cars, computers, equipment etc., examination time scheduling and several other interesting problems.

In this assignment, a special case of a symmetric Euclidean TSP is used to study the various optimization algorithms. The problem solved will be, given *n* cities and picking an arbitrary city to start from, there exists $\frac{(n-1)!}{2}$ possibilities for a round-trip. The factor 2 arises from the fact that we don't care about the direction in which we are traveling. The capitals of 48 US states, which excludes Alaska and Hawaii, are used to find the optimal round trip around the nation. The total mileage needed to optimally traverse the capital cities is 10,627.75 mi {4} and the figure below shows the optimal route;

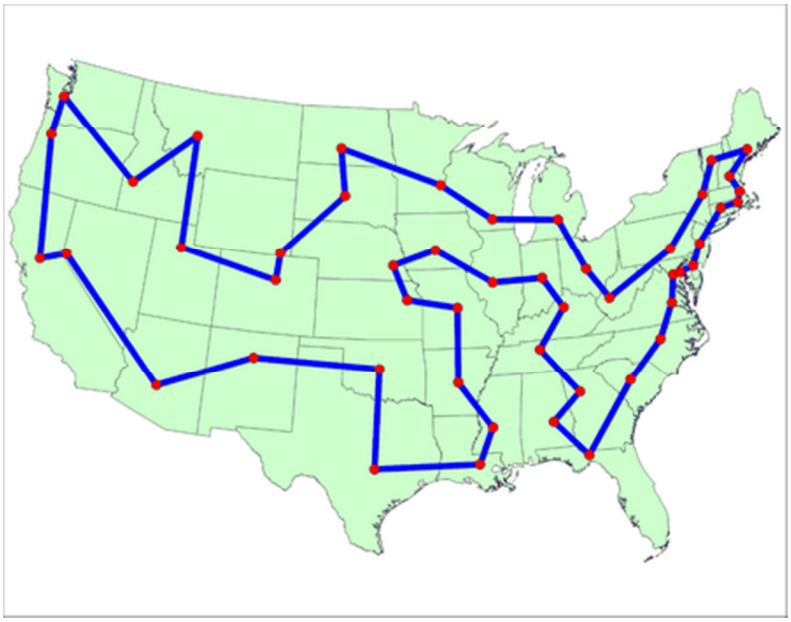

Figure 2: Traveling salesman problem, optimal route around 48 US states.

This problem is best suited for the simulated annealing (SA) algorithm {5}. Due to the "annealing" or varying temperature the chances of getting stuck in a local minima is less likely than the "hill climbing" approach where the "temperature" is kept at 0, i.e., accepting new tours if and only if it's better than the existing tour. In SA an inferior tour is accepted temporarily because it might be the stepping stone that leads to the global minimum. This will be further explored in the results section. Finally, although this is a minimization problem, it can be easily transformed to a maximization problem by negating the cost function.

OneMax Function:

The goal of the OneMax problem is to maximize the number of 1's in a bitstring. This can be summarized by defining a function which tries to maximize $\sum_{i=0}^{n} x_i$ where, *x* is a bit vector of length *n*. The max will be obtained $x_i = 1 \; \forall \; i$ i. This is a highly oscillatory function and the objective function only provides an indication of the number of correct bits in a candidate string and not the positions of the correct bits. Although this is a simple maximization problem, applications in chromosome sequence generation have been reported {6}. This problem could be best suited for genetic algorithms (GA) and the convergence can be accelerated by MIMIC. The function representation is trivial and discrete is not presented for this report.

Optimization Algorithms

Randomized Hill Climbing (RHC):

The MATLAB function ***patternsearch*** which is based on a hill climbing algorithm was used to study the above three problems. ***Patternsearch*** attempts to locate a better point with a lower objective function value than the current point and if it is able to find a better point then that becomes the current point and no polling is done for that iteration. If the search does not find a better point, ***patternsearch*** performs a poll. Two different poll schemes were tested for ***patternsearch*** and results were reported. The first poll stops when a neighbor with a better fitness function is values is found (firstpoll) and the other evaluates all neighbors and takes the best one (complete poll) {7}.

Genetic Algorithm (GA):

The MATLAB function ***ga*** was used to implement a genetic algorithm to the above three problems. Various population sizes and initial parameters for mutations of Gaussian noise and crossover of 80% of old population replaced by children were used. The best two individuals were kept among generations.

Simulated Annealing (SA):

The MATLAB function ***anneal*** was used to study the above three problems. This algorithm is sensitive to the initial temperature and the perturbation functions used. Doing an entire parameter search for the best initial time and perturbation functions is beyond the scope of this assignment and hence the best parameters found by trial and error were used.

MIMIC:

The implementation from **ABAGAIL** was used to study the three problems above. The population size was chosen to be 200 {8} and 100 iterations were used by default.

Results and Discussion

In this section, the results for the three problems studied are presented. Since all the algorithms used in this assignment require randomized initializations, each of them was run several times and the means from all these runs were used to evaluate the performance. The various settings and parameters used for these algorithms are also discussed in this section.

Rastragin's Function:

All algorithms were run 20 times except **MIMIC** which was run 50 times. A randomization function was used to generate the initial points in the interval [-20, 20]. GA was run for a varying population size of 5 to 50. In order to evaluate the performance of the algorithms, time and the error associated with each was monitored and analyzed. Error was computed by the distance between the true minimal function value of zero and the final minimum found by the algorithm. The total number of function evaluations was compared to study the efficiency of each of these algorithms. Note that since each algorithm was run multiple times, the average function evaluations across all runs were computed. Presented below is a summary of the results from the various algorithms;

Algorithm	Time (s)	Error (ε)	Function Evaluations
RHC – first poll	16.4057	1.6894e-18	1488.6
RHC – complete poll	16.6713	1.7764e-16	1520.4
GA	30.6193	3.2535	5777.975
SA	6.8392	0.0018944	11633.3
MIMIC	181.1694	1.1832	20010.4

Table 1: Rastrigin's function results for the four algorithms studied.

From the above results, it can be concluded that the RHC and SA algorithms outperform the genetic algorithms. While SA has the fastest run time, RHC has the lowest error but only marginally. The RHC algorithm is much more efficient compared to the SA algorithm since it needs significantly fewer function evaluations. This could become important for complex functions as this will directly impact the run times. The two variations of the RHC function perform similarly.

Since the Rasrigin's function has several local minima, algorithms that have the capability of examining points far away from the current basin of interest have a better chance of finding the global minimum. SA starts out with a high temperature which helps it cover a larger domain space and slowly reduces its search area (cools) as it converges to the global minimum. This property helps SA find the global minimum where several local minima exists. Generally, RHC will find the global minimum when the initialization happens to be in the right region of the search space but the ***patternsearch*** algorithm used in MATLAB allows expansion of the search radius and hence the RHC algorithm converges to the global minimum. GA and MIMIC perform poorly for the Rastragin's problem. GA uses non-linear bit-

transformations and due to the smoothness of the Rastrigin's function the performance is severely hampered. Variations in population do help bring the error down (Figure 3) but does not reduce to the extent of the other algorithms. Although, varying the population size improves the results, the number of function evaluations (Figure 3) required increases and thus the total run time also increases. An interesting observation for GA and MIMIC is that, once MIMIC falls into the valley of interest of local minima it finds the local minimum precisely whereas GA finds points scattered around the local minima (Figure 4). This can be attributed to the fact that MIMIC interpolates the probability distribution thus accounting for the smoothness of the Rastrigin's function.

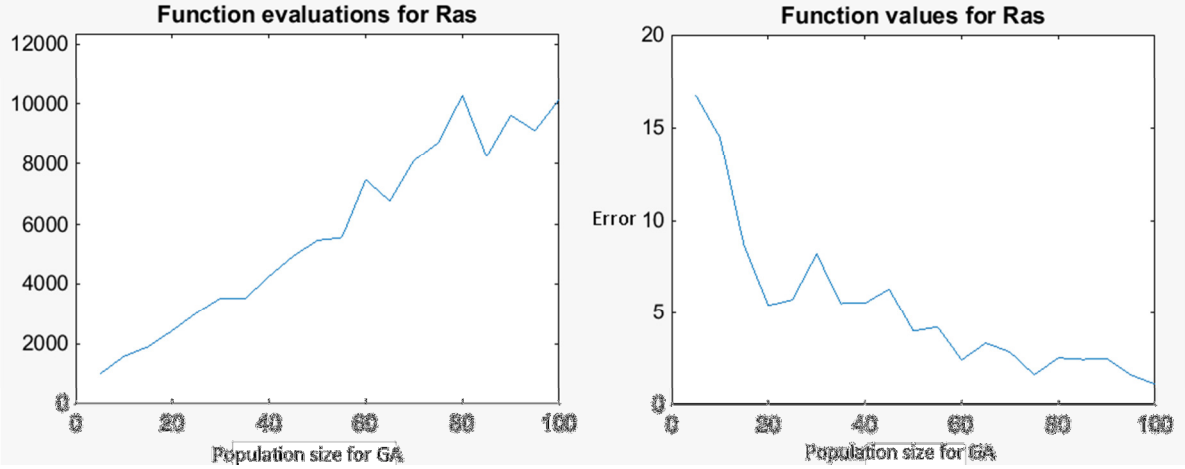

Figure 3: GA for Rastrigin's function, function evaluation and error as a function of population.

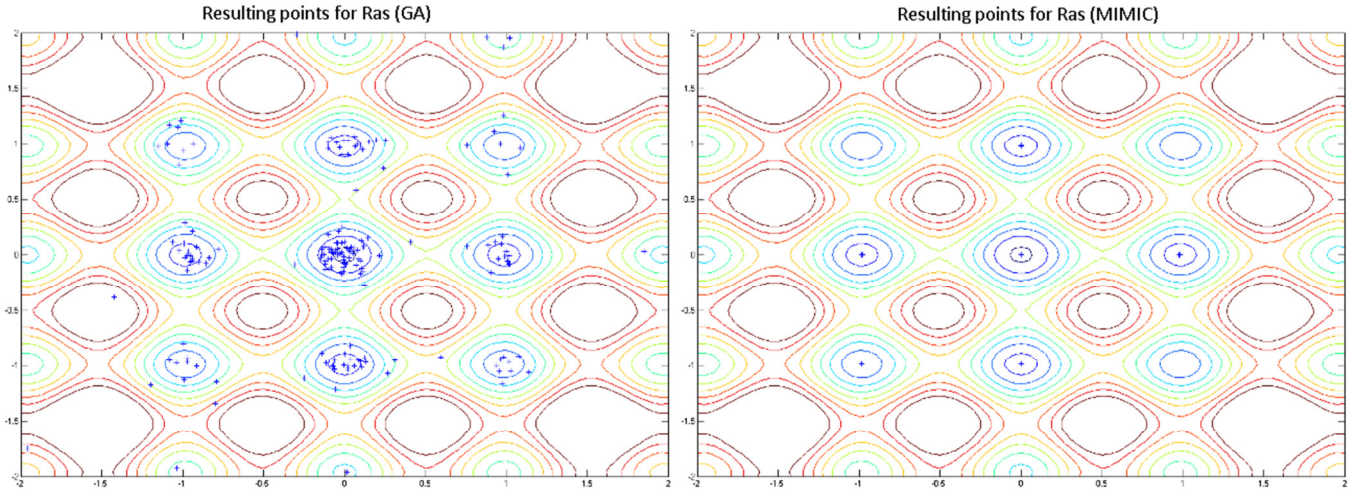

Figure 4: Best points for Rastrigin's function for GA and MIMIC.

Traveling Salesman Problem (TSP):

All algorithms were run 10 times, except MIMIC which was run 5 times. GA was run for a varying population size of 5 to 50. In order to evaluate the performance of the algorithms, time and the error associated with each was monitored and analyzed. Error was computed by the distance between the true minimal function value given by the optimal tour and the final minimum found by the algorithm. The total number of function evaluations was compared to study the efficiency of each of these algorithms. Note that for TSP the run times are extremely long and takes several iterations to reach the optimum solution. In order get some results in reasonable time the solution was run only 10 times and the convergence was modified to complete the runs. This may not yield the global minimum solution but provides a framework to compare the algorithms to the same point of convergence. Since each algorithm was run multiple times, the average function evaluations across all runs were computed. Presented below is a summary of the results from the various algorithms;

Algorithm	Time (s)	Error (ε)	Function Evaluations
RHC – first poll	983.6617	17474.4281	4562.2
RHC – complete poll	1057.79	16589.3421	4729.5
GA	690.8	47786.9	3937.9832
SA	36.216	13299.91	6217.2
MIMIC	331.387	28981.4512	67654.9

Table 2: Traveling salesman problem results for the four algorithms studied.

The algorithms are started by picking an arbitrary initial tour from a set of all valid tours. From the initial tour a set of random neighboring tours are selected to evaluate if they are better. There are many tours and testing every possible solution is not feasible but the algorithms are designed to reach a, if not the global optimum, is at least good enough. This was important in implementing the algorithm to get results in reasonable times. From the above results, it can be concluded that SA does the best for the TSP problem. It took the least time and had the lowest error compared to all other algorithms. This could be attributed to the fact that while considering a tour that is worse than the current tour, the worse tour is temporarily accepted because this might be the stepping stone to getting out of a local minimum and ultimately get closer to the global minimum. The temperature is usually pretty high at the beginning of the annealing process, so that initially more tours are accepted, even the bad ones and over time as the temperature is lower only new tours that improve upon the current tours is accepted.

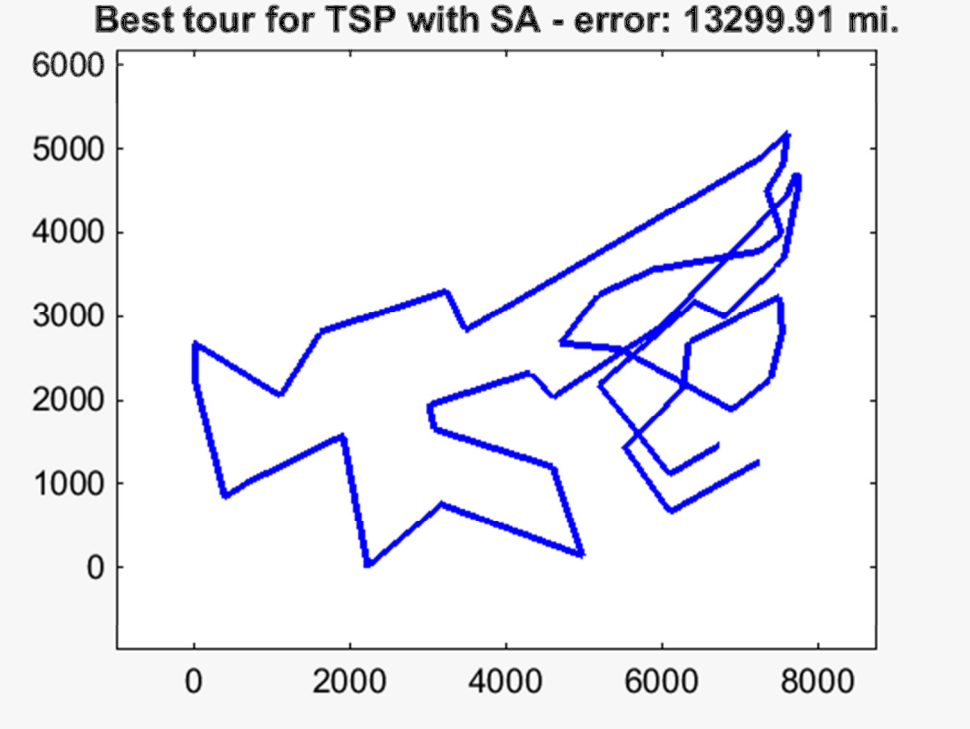

Figure 5: Best route obtained for TSP problem from SA.

The next best algorithm is the RHC which explores a larger search space and hence is able to converge to a global minimum. MIMIC and GA perform poorly compared to the other algorithms and this could be due to the interpolations carried out in the cross-over and mutations operations. It must be noted that with increasing population the error drops and the number of function evaluations increase (Figure 6). Presented in Figure 5 is the optimum route generated by SA. The best routes for other algorithms can be found in the plots directory of the assignment. Upon further investigation, several variations of SA and GA are available that solve the TSP very well {9}.

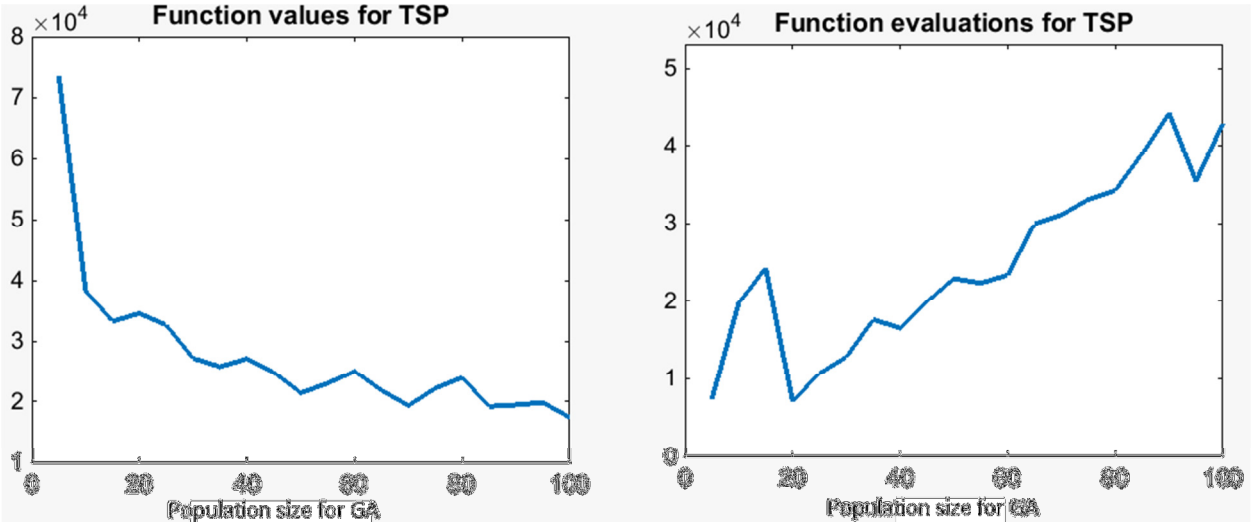

Figure 6: GA for TSP, error and function evaluations as a function of population.

One-Max Function:

All algorithms were run 100 times, except MIMIC which was run 10 times. GA was run for a varying population size of 5 to 50. In order to evaluate the performance of the algorithms, time and the error associated with each was monitored and analyzed. Error was computed by the distance between the true minimal function value of 80 and the final maximum found by the algorithm. The total number of function evaluations was compared to study the efficiency of each of these algorithms. Note that since each algorithm was run multiple times, the average function evaluations across all runs were computed. Presented below is a summary of the results from the various algorithms;

Algorithm	Time (s)	Error (ε)	Function Evaluations
RHC	0.09	10.52	20000
GA	0.08	17.28	20000
SA	4.9	31.79	60000
MIMIC	13.8	3.1	50000

Table 3: One-max function results for the four algorithms studied.

As opposed to the other algorithms, MIMIC analyzes the global structure of the optimization landscape and guides the randomized search through the solution space. Due to the discrete nature of the one-max function MIMIC is able to perform the best compared to other algorithms as the estimate structure is refined as the search progresses and thus hones in on the global maximum. Although, MIMIC was the most accurate among the algorithms it was also the slowest and this stems from the fact that the underlying global structure is computed every iteration. SA had the worst performance since the search space is guided only by the surrounding points and has no information regarding the underlying structure of the function and since the one-max function is highly oscillatory SA fails to converge to the global maximum. For similar reasons, RHC and GA converge to local maxima but perform better than SA.

Neural Network Study

In assignment 1, for the car evaluation data set, a classification accuracy of 93.5 % was shown. By running this data set on a 7 units and 1 hidden layer neural network a classification rate of 78.9% was achieved using the GA for specifying the weights. The sum of squared errors with the regularization term was used to study the performance of the neural network and also was used to keep the solution from blowing-up. The regularization is the addition of the norm weights of the units. The car data set has 6 dimensions and hence the neural network problem solved was a $6 * 7 + 7 * 1 = 49$ dimensional problem. The weights were initialized randomly and the optimization algorithms were run 10 times. The mean results are presented in table 4. The accuracy of the neural network was studies by applying it to a test set of the car data set.

Algorithm	Time (s)	% Correctly Classified Instances	Function Evaluations	Error
RHC – first poll	983.1	64.3	5124.56	0.3572
GA	1028.98	78.9	3689.72	0.2118
SA	654.23	62.8	3984.94	0.3715

Table 4: Neural Network results for the algorithms studied.

GA performed the best in terms of % of instances classified correctly but took the longest time. SA took the shortest time but had misclassified 37.15% of the instances. RHC had the highest function evaluations. These tests were run without cross-validation and hence the poor performance on the test set indicates that the classification problem suffered from overfitting. A more detailed application where the data set is cross-validated will give more reliable results. Also, increasing the dimensionality of the neural network will improve the results.

Conclusion

In this report four randomized optimization algorithms were applied to three functions which pose different challenges to the optimizer.

The Rastrigin's function although smooth has several local minima and hence requires the optimizer to be more robust and reliable so it doesn't get stuck in these local minima and continues to find the global minima. SA and RHC algorithms performed the best for this problem. This can be attributed to the temperature schedule for SA where a larger area or the function is covered initially to eliminate local minima as much as possible while honing in on the global minimum. In the case of RHC the widening of the search area helps it perform a similar function.

The TSP was challenging to all algorithms due to the complexity of the search space. SA had the best result again due to its approach in handling the search space. Accepting a bad solution temporarily and using it as a stepping stone to find the global minima proved critical in its performance. Other algorithms got stuck in local minima and hence fared badly compared to SA.

The highly oscillatory nature of one-max function helped MIMIC perform well for this problem. Understanding the global structure of the function and using it to converge to the global minimum proved vital in its success. GA worked reasonably well for this problem as well.

Finally, using weights derived from these optimization algorithms was applied to a neural network which was used to classify the car data set from assignment 1. The lack of proper cross-validation and restriction on the structure of the neural network hindered its successful application. Overfitting was clearly evident from the results.

References

{1} http://www.mathworks.com/help/gads/example-rastrigins-function.html#f10136

{2} http://www.csd.uoc.gr/~hy583/papers/ch11.pdf

{3} http://oai.cwi.nl/oai/asset/21825/21825A.pdf

{4} http://support.sas.com/documentation/cdl/en/ornoaug/65289/HTML/default/viewer.htm#ornoaug_optnet_examples07.htm

{5} http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/

{6} http://pontasdamadrugada.blogspot.com/2014/02/compact-genetic-algorithm-solve-onemax.html

{7} http://www.mathworks.com/help/gads/direct-search.html

{8} http://www.cc.gatech.edu/~isbell/papers/isbell-mimic-nips-1997.pdf

{9} http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/