CS-7641 Machine Learning: Assignment 2

Randomized Optimization

Chenzi Wang
cwang493@gatech.edu
October 15, 2015

1. Optimization problems

The purpose of this assignment is to observe and understand the utility of a variety of random search algorithms, specifically Randomized Hill Climbing (RHB), Simulated Annealing (SA), a Genetic Algorithm (GA), and MIMIC. To demonstrate their functionality, three optimization problems were chosen with the intention of showing the strengths and weaknesses of each algorithm. The three I decided on are a Rastrigin function, the Traveling-Salesman-Problem (TSP), and the One-Max-Function. A neural network will also be used to evaluate them, except for MIMIC. The first two, Rastrigin's function and the TSP, are actually minimization problems; however, they can be easily changed to optimization/maximization problems by changing the sign of their fitness functions such that they are suitable for the purposes of this assignment.

1.1 Rastrigin Function

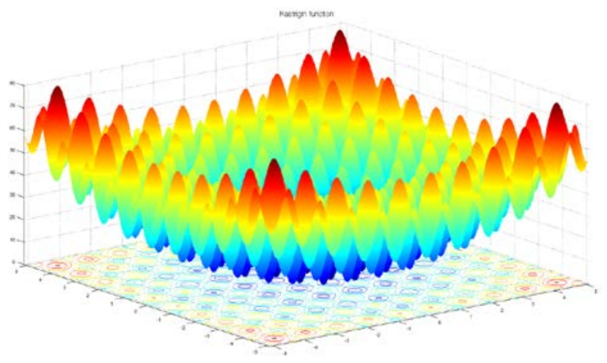
The Rastrigin Function is a non-convex function characteristically used since its inception as a test problem for optimization algorithms.[1] The function comes from the addition of a periodic function and a parabola:[2,3]

$$f(x_1 \dots x_n) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$$

It has the virtue of having infinite local minimums but only one global minimum at [0, 0] in the [x, y] plane (it is multimodal). This causes an increased probability that an optimizer algorithm will fall into a local minimum, especially if it relies heavily on gradient descent methods. Consequently, it has often been used to determine the accuracy of genetic algorithms.[4] Algorithms that test well with a Rastrigin function would work well with other applications that have local minimums, which are common in many real-life problems which originate in physics, chemistry, engineering, and mathematics.[5] Though this may be seen as a continuous problem, representing floats as 32 bit binary vectors permits this function to be treated as a discreet problem. The Rastrigin function I have chosen for analysis in this assignment is (see Figure 1 for a graphical representation):

$$(x, y) = 20 + 3(x^2 + y^2) - 10(\cos(2\pi x) + \cos(2\pi y))$$

[1] Rastrigin, L. A. "Systems of Extremal Control." (1974).
[2] http://www.cs.unm.edu/~neal.holts/dga/benchmarkFunction/rastrigin.html
[3] http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page2607.htm
[4] http://www.mathworks.com/help/gads/example-rastrigins-function.html
[5] Yang, Xin-She. *Swarm Intelligence and Bio-inspired Computation: Theory and Applications*. Amsterdam: Elsevier, 2013.



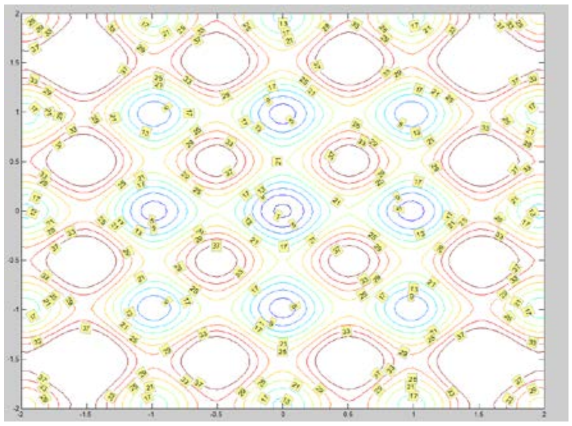


Figure 1: Rastrigin's function: (left) surface plot, (right) contour plot

1.2 The Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) consists of determining the least costly round-trip route for a salesman who needs to visit a number of scattered cities each one time. The cost can be a number of things, including travel time, price of tickets, and distance. For the purpose of this assignment, the measurable cost is distance traveled and I have chosen the capitals of the 48 contiguous states of the USA. The TSP problem is considered difficult to solve because, if there is a way to break the problem up into smaller component problems, each component problem will be at least as complex as the original problem. Such a problem is termed NP-hard in combinational optimization.[6] The simplest and most computationally demanding method is to try all combinations of routes. This is intensive because that means with N number of cities, there are (N-1)! possibilities. That would me for the 48 cities I have chosen for this assignment, there are $(48-1)! / 2 = 1.3 \cdot 10^{59}$ possible routes to evaluate. The number of factorial possibilities is divided by two because it does not matter which direction is traveled around the route since distance is the weighing factor (it is a symmetric TSP).[7] Finding an algorithm that works best for this problem is difficult because small changes in parameters (i.e. locations of cities) can give completely different optimal routes. It is unlikely that a greedy algorithm will accurately determine an optimal route from one set of parameters to another. There is no universal solution for the general case of a TSP.[7] The TSP used for this assignment has a known shortest route of 33523.7 miles.[8,9,10] The optimal route is shown in Figure 2. The traveling salesman example is valid for many real-life problems such as for a tourist who wants to visit a number of landmarks in the shortest time possible, the quickest route for assembly on a manufacturing line, shipping packages, quickest travel routes for laser making microchips, and for use in determining GPS travel routes (i.e. shortest distance given a number of intersections).

[6] http://mathworld.wolfram.com/TravelingSalesmanProblem.html
[7] http://www.math.uwaterloo.ca/tsp/problem/index.html
[8] Skiena, Steven S. *The Algorithm Design Manual*. Santa Clara, CA: TELOS--the Electronic Library of Science, 1998.
[9] Liu, Jianjun, Yuan Li, Xinrui Wang, Yuan Wen, and Tingying Zhou. "A Two-phase Methodology Heuristic IInsertion Algorthm for TSP." Transactions on Computer Science and Technology 2.4 (2013): 62-68.
[10] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

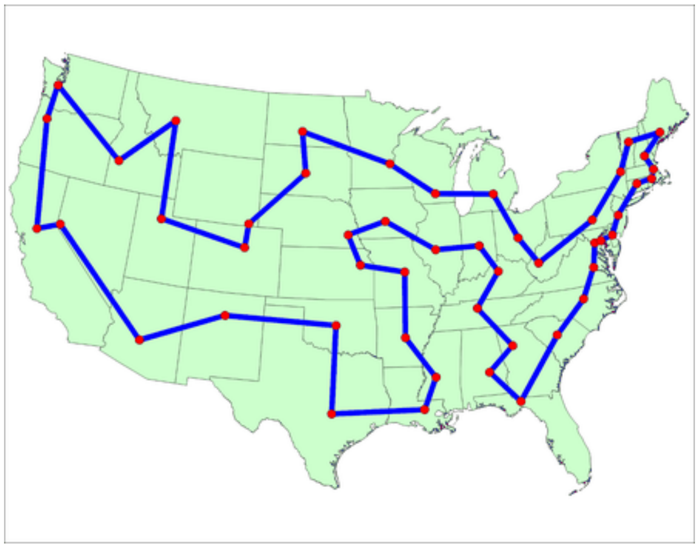


Figure 2: TSP problem - the optimal route through the capitals of the 48 states making up the contiguous USA.[11]

1.3 One-Max Problem

The One-max problem is a simple problem in that the goal is to maximize the number of ones in a bitstring. Taking a bitstring x with a length of n, the problem which an algorithm must attempt to solve is to determine a bitstring that maximizes the following discreet function:[12]

$$f(x_i \dots x_n) = \sum_{i=1}^{n} x_i \quad where\ x_i \in \{0,1\}$$

The optimal answer for a bitstring of length n may seem obvious to a person solving this problem since the optimal bitstring is one where all of the bits have been set to one. For example, the maximum value for a bitstring with length of 8 bits would be 8. However, this problem is not quite so simple for an algorithm because the value oscillates with each combination of ones and zeroes, as can be seen in Figure 3. It should be noted that a small change in parameters will have the same small effect on the result from an algorithm (i.e. changing a bit x_i from 1 to zero will subtract 1 from the result). The One-max problem has been shown to functionally test interactive and human-genetic based algorithms.[13]

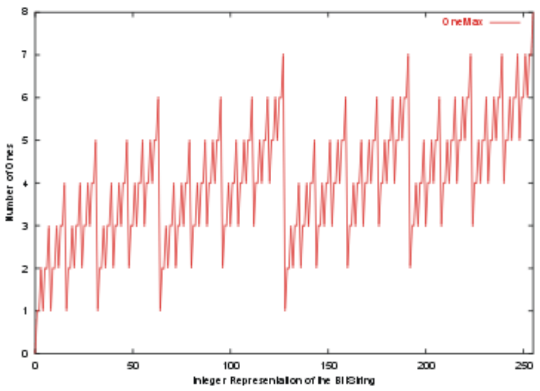


Figure 3: One-Max function of a bitstring with a length of 8 bits.[12]

[11]http://support.sas.com/documentation/cdl/en/ornoaug/65289/HTML/default/viewer.htm#ornoaug_optnet_examples07.htm
[12] http://tracer.lcc.uma.es/problems/onemax/onemax.html
[13] http://www.academia.edu/2974592/Interactive_one-max_problem_allows_to_compare_the_performance_of_interactive_and_human-based_genetic_algorithms

2 Optimization Algorithms

The algorithms to be compared using these problems as well as a neural network are:

Randomized Hill-Climbing (RHC): This is an algorithm that operates by initially setting an arbitrary solution, then altering the solution incrementally, changing one parameter at a time, to determine if a better solution can be made. This is repeated until no further improvements can be made. To implement this algorithm, the MATLAB function *patternsearch* was utilized using random starting points. *patternsearch* uses the standard generic hill-climbing algorithm but also utilizes an additional function that increases or decreases the search radius at every step in iteration depending on whether there is a neighbor with a better fitness function value. In other words, it polls the neighbors to continually resize its search radius. I used two different polling methods for testing, with the difference being the rule for determining how many neighbors to poll before moving on to the next iteration step. The first stops polling its nearest neighbors and moves on to the next step once a neighbor is found with a better fitness function value. The other method evaluates the fitness function values of the all neighbors and takes the best one before moving on to the next step (complete polling).

Genetic Algorithm (GA): The genetic algorithm is a search algorithm designed to mimic the process of natural selection. To implement this algorithm, the MATLAB function *ga* was used with different population sizes. Mutation is an operator used to maintain diversity among each set of "chromosomes" in a generation of "chromosomes." Mutation does this by altering a parameter of some of the chromosomes of each generation; the probability for a parameter to mutate is set by the operator. For this assignment, Gaussian mutation is used. Meaning that if a mutated parameter falls outside user set limits, the parameter is cut to that limit. Crossover is the process of taking more than one parent solution to produce a child solution from them. For this assignment, 80% crossover is used, meaning 80% of parent solutions will be replaced by child solutions. The best 2 values (individuals) are passed to the next step from each generation.[14]

Simulated Annealing (SA): The simulated annealing algorithm was designed such that it mimics the annealing process in metallurgy, where a metal is brought to a high temperature then slowly cooled to decrease defects. At each iteration step in the algorithm, a new point is generated randomly. The algorithm then accepts all the points which improve the function value. However, it does accept some points that do not improve the function value. By doing so, the algorithm acts to deter falling into any local maxima/minima. The difference between the new point and previous points and/or from the limits of the search is determined from a probability distribution scaled proportionally to the temperature.[15] Consequently, SA is sensitive to the initial temperature and the perturbation functions used. This was unsurprisingly found to be the case when running an SA algorithm for the problems in this assignment. One could search for the best parameters for each problem online, but I considered such a task to go beyond the purpose of this assignment. Instead, I found the best values I could within reason via trial and error. Looking at the values I found which worked best, I think it is most likely that it was specifically the aforementioned scale/ratio between the perturbations and the temperature which had the strong effect.

Mutual-Information-Maximizing Input Clustering (MIMIC): This algorithm attempts to alter later search iterations by communicating lessons (discovered mistakes and pitfalls such as local maxima/minima)

[14] http://www.obitko.com/tutorials/
[15] http://www.mathworks.com/discovery/simulated-annealing.html

about the fitness function from previous search iterations.[16] To implement this algorithm, the MIMIC java code for each problem from ABAGAIL was used. As per the experiments in [16], a population sample size of 200 was chosen as well as 100 iterations.

For the One max problem, code from ABAGAIL was used for all algorithms.
Each time these algorithms are run on each problem, a new result value is produced. This is because they are all randomly initialized. To attain robust function values for analysis, these algorithms were run multiple times each with the average of these output values reported in the present assignment.

3 Results

3.1 The Rastrigin Function Results

For this problem, the algorithms were run 10 times. This was due to variation in results for the GA and MIMIC algorithms. The population for GA was run for sizes varying from 5 to 50. The average results of these runs can be found in Table 1. The error is the distance from the global minimum, which is known to be at [0, 0]. The points being tested were constrained within a two dimensional plane $[-20, 20] \times [-20, 20]$.

It is clear from the results that RHC and SA provide more accurate answers than the other algorithms, both have essentially no error. For this problem, there is no difference between the accuracy of both forms of RHC polling since they both produced completely accurate results. The complete polling method, however, has a slightly shorter computing time and less functional evaluations to reach its result. SA, on the other hand, outperforms RHC in both clock time (27% less) and in the number of function evaluations (44% less). Because of the multimodal nature of the problem, the best algorithms for use in its evaluation need to continually assess point(s) away from its current best point at each iteration step to deter falling into the multitude of local minima. SA accomplishes this by occasionally allowing new points for analysis that do not give the function a more optimal value. This increases the probability that a point will be evaluated near the global minimum leading to optimization towards it. RHC in its simplest form would not likely perform well in this problem because it is highly dependent on its initiation point, which is random, since it only examines points in its immediate surroundings. However, the form of RHC used in this assignment did well due to its utilization of neighbor polling to expand its search radius. I would say the <u>algorithm that performed best for this problem was SA</u> due to its accuracy, fastest computational time, and because it has fewer evaluations than RHC, which had equivalent accuracy. Compared to the other algorithms, GA performed poorly. It produced results with a higher error than any other algorithm. Its clock time is also more than RHC or SA. It produces its results with fewer function evaluations, but this does little to redeem itself. It should be noted, however, that increasing the population size did have the direct effect of decreasing error, though this comes with the price of increased clock time and function evaluations. This is illustrated in Figure 4. Figure 5 shows that a key issue with the GA algorithm is that its evaluations are prone to falling into local minima. The MIMIC algorithm does not do as well as RHC or SA, but it is superior to GA for this problem. It can be seen from Figure 5 that, like GA, MIMIC evaluations are also prone to falling into local minima. However, a key difference between the two can also be observed. The values of GA seem to be scattered around the minima whereas the values attained through MIMIC are seen to arrive at the precise minima of whatever minima it fell into. Consequently when the minimum it fell into was the global minimum, these runs gave absolutely accurate results. This might be because the Rastrigin function is a smooth continuous function that has been made discreet for the present purposes. The fact that MIMIC

[16] http://www.cc.gatech.edu/~isbell/papers/isbell-mimic-nips-1997.pdf

interpolates the probability distribution may be why it accurately finds minima. GA, in contrast, uses non-linear mutations to alter its chromosomes during iterations. Such non-linear mutations are likely responsible for the algorithm not finding the exact center of the minima. MIMIC does, however, have the highest clock time (650% longer than GA) of all the algorithms by a large degree, possibly showing that it is computationally demanding. I say possibly because MIMIC was performed using JAVA whereas the rest were evaluated using MATLAB.

Table 1: Rastrigin Function Results

Algorithm	Clock Time	Func. Eval.'s	Error
RHC - first poll	20.7	29672	0
RHC - complete poll	19.1	29406	0
GA	27.6	2885	9.72
SA	5.5	12984	0.001
MIMIC	179	20000	1.22

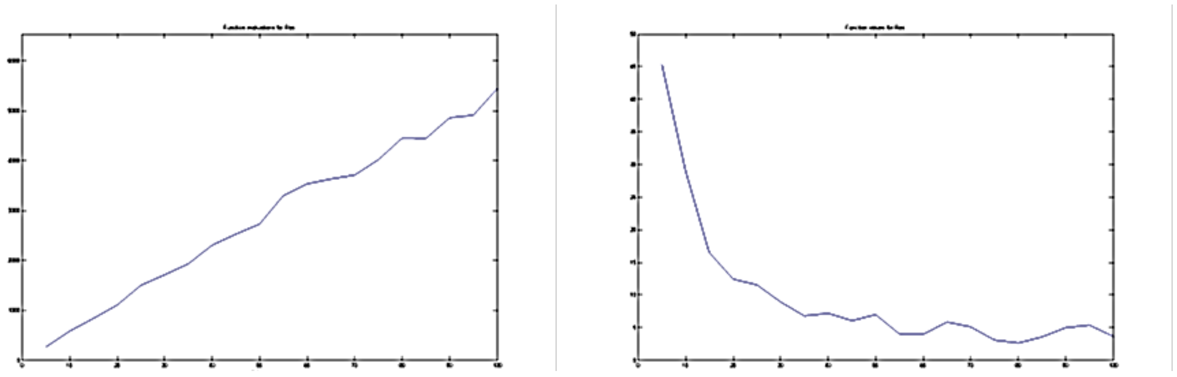


Figure 4: GA results for the Rastrigin function - (left) error for different populations and (right) number of function evaluations

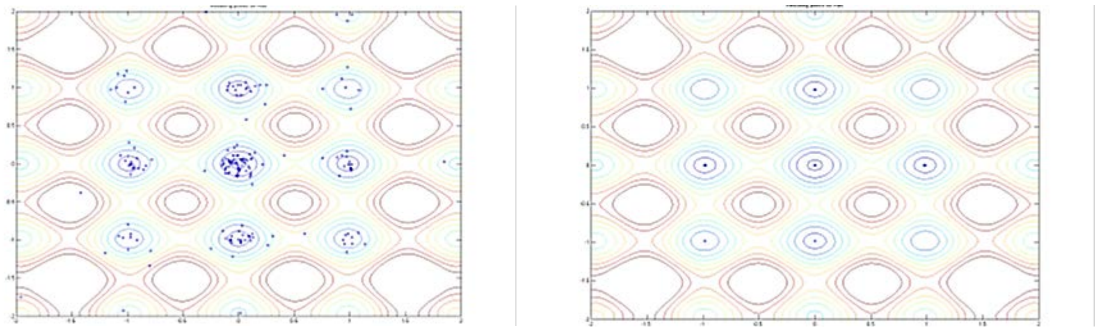


Figure 5: Best points for the Rastrigin function from (left) GA and (right) MIMIC

3.2 Traveling Salesman Problem (TSP) Results

For this problem, the algorithms were run 5 times each and GA was run for population sizes between 5 and 50 (because there are 48 data points, a population size of 50 utilized the full population). The average results of these runs can be found in Table 2. As mentioned previously, the known optimal route length for this problem is 33523.7 miles. To come up with a result, the algorithms output travel courses, which were obtained by sorting the vector of locations. Each location had latitudinal and longitudinal data that were used equivocally as x and y coordinates. A simple function was then implemented to calculate the distance between one point in the vector and its subsequent point. The algorithms consequently function by attempting to optimize the order of the vector of locations such that the sum total of distances between each point in the vector and its subsequent point is minimized. A possible error in this approach is that it does not permit locations to be passed through twice; however, there is strong evidence that this capability would not improve results.[8]

All algorithms produced results ~50% or greater than the known optimal value. This problem does not involve a smooth/linear function. Because this problem is NP hard, one would not expect an algorithm that is highly dependent on the random initiation point or that uses interpolation or lessons from previous iterations to perform well. Consequently, it is not surprising that GA and MIMIC have the poorest accuracy of the algorithms, with GA having the worst accuracy of the two. The output value of GA did improve with increased population (see Figure 4 for the equivalent behavior seen with the Rastrigin function), but does not surpass any other algorithm for accuracy even with the maximum population size used. It should be noted, however, that there may exist variants of this problem where the function of a GA would improve if the algorithm were to more heavily utilize mutation and cross-over operations. The RHC and SA algorithms produced outputs with similar accuracy, the highest of the group of algorithms. This is likely because these algorithms perform their evaluation by looking at neighbors covering a large search space; their function does not depend on interpolation. In this case, the RHC that polled all of its neighbors did outperform the first order polling method, achieving 3% better accuracy with fewer evaluations and clock time. However, though the accuracy of SA is between the accuracies of the two RHC methods, I would say <u>SA has the best performance of all the algorithms</u> because it achieved an equivalent accuracy to RHC in a much shorter clock time. SA did have more evaluations than RHC, but, as shown be the clock time, an SA evaluation was much faster than a RHC evaluation by a large degree, meaning it is less computationally demanding. The reason for this difference in speed is likely because the form of RHC used presently evaluates a large number of neighbors for each iteration looking for a better neighbor before moving on to the next step. On the other hand, SA does not look specifically at neighbors but at a limited number of random points at each iteration step. This also means that the random starting point for SA is not as important as in RHC. This did not equate to finding its best value within a smaller number of evaluations, but it does mean that the evaluations performed at each step were much faster. A comparison of the routes obtained from the worst (GA) and best (SA) performing algorithms for this problem can be found in Figure 6.

Table 2: TSP Results

Algorithm	Clock Time	Func. Eval.'s	Avg. Value (mi.)	Error (mi.)	Error, %
RHC - first poll	3475	4615	50942.2	17418.5	52
RHC - complete poll	3371	4516	49933.8	16410.1	49
GA	692	3938	81315.8	47792.1	143
SA	130	6689	50386.3	16862.6	50
MIMIC	332	100000	66111.9	32588.2	97

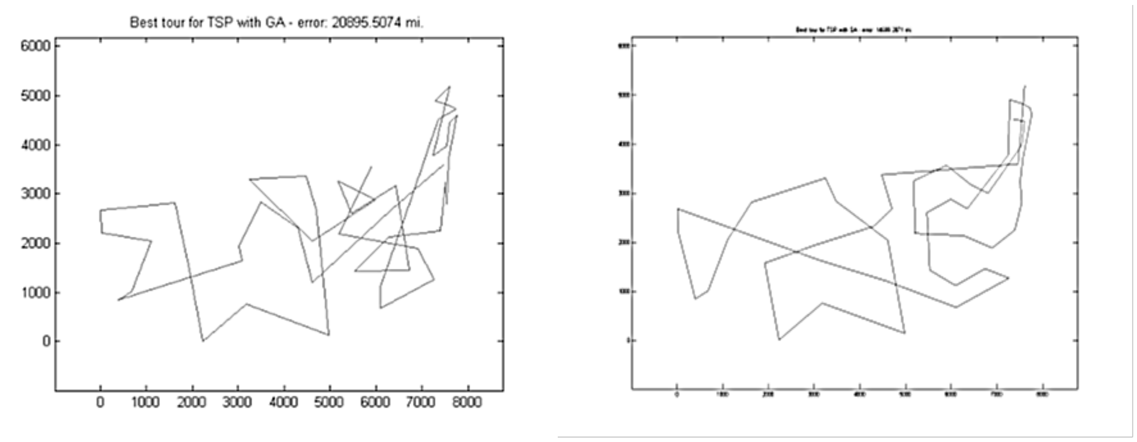


Figure 6: Best routes obtained for TSP problem using (left) GA and (right) SA

3.3 One-Max Problem Results

For this problem, the algorithms were run 10 times to produce the average results found in Table 3. For optimization of the one-max function, an 80 bit long bitstring was chosen. Accordingly, the optimal value for the function is 80. Also, 80 bits gives 2^{80} = 1.2 10^{24} possible combinations.

<u>MIMIC outperformed the other algorithms on this problem</u> due to its far superior accuracy; I feel that the facts it has the longest clock time and higher evaluations are not enough to negate the accuracy of its result. Since all algorithms in this program were performed through JAVA, it is clear here that MIMIC is more computationally demanding than the other algorithms, having the most clock time per evaluation. I feel I can say with confidence that its accuracy is due to its use of interpolation and learning from previous iterations (it will act to interpolate previously evaluated points), which is very well suited for this function given its shape (though discreet, it approximates a continuous function - see Figure 3). The other algorithms do not operate in the same manner and, consequently, cannot match the accuracy of MIMIC. This is because they are searching without guidance. In this case, RHC had the second highest accuracy because of its added capability in the variant used here to search neighbors for optimal values, consequently giving it a large search area. SA has the worst performance in accuracy in this problem because it checks random points (of which there many) rather than looking more at neighbors, which makes it inefficient in this case and reducing the probability it will find the optimal value. The accuracy of GA comes in between those of RHC and SA due to the ability to increase its search area by increasing its population size. However, the mutation operation in GA may actually be working against it, decreasing its efficiency in the same manner as the random points in SA, though only the points creating a better value will be allowed to pass on.

Table 3: One-max problem results

Algorithm	Clock Time	Func. Eval.'s	Avg. Value	Error	Error, %
RHC - first poll	0.08	20000	68.9	11.1	14
GA	0.06	20000	61.1	18.9	24
SA	4.4	60000	47.8	32.2	40
MIMIC	16.2	50000	77.2	2.8	4

3.4 Neural Network Results

I decided I needed to change to a different problem from the ones I chose in the previous assignment. The problems I had chosen for the first assignment involved stock data prediction. Upon reevaluation, it is obvious this choice did not meet the requirements of this assignment. With the better understanding I now have, it is obvious to me that my previous problem was made up of continuous values and was a problem of regression rather than of discreet optimization necessary for this assignment. Rather than continue with a previous problem which would only produce results on which I cannot make meaningful analysis, I have decided to choose a different problem more aligned to the purpose of the assignment. This new problem and neural network was constructed using MATLAB.

The neural network analyzed here is one in which the algorithm attempts to recognize cancer cells from characteristics observed in microscopic images. This problem attempts to accurately determine whether a small number of cells removed from a tumor through a minimally invasive surgical process can be used to determine whether the tumor is malignant or benign. The data used contains 20 measured characteristics along with the diagnosis (malignant/benign). If algorithms using the principles of machine learning can be used effectively, a diagnosis can then be reached without the need of risky invasive surgery or the use of x-rays, which are not very accurate and expose a patient to radiation. If the principles of machine learning can be used effectively for such a problem, it is likely they can be used in other areas of medicine to achieve a diagnosis from measureable characteristics (even such things as body temperature and symptoms) without performing risky or costly investigative procedures. Such a problem is primarily one of pattern recognition.

The neural network algorithm I am using has 1 hidden layer of 7 units. During training, it utilizes the minimization of the Sum of Squared Distances (SSD) of the error with a regularization term to prevent the solution from blowing-up, which may have been a risk for RHC due to its method of searching. The regularization term used is the sum of the normalized weights of the units. The backpropagation method of coming up with weights, which is replaced by optimizing algorithms for this assignment, produces recognition rates greater than 95%. The function being optimized by the optimization algorithms is complex due to there being 20 characteristics, 7 units, and 1 hidden layer: 20*7 + 7*1 = 147 dimensional function. To use this neural network algorithm for the purpose of this assignment, the weights were initiated with random values to be updated by the optimizing algorithms.

Each algorithm (RHC, GA, and SA) was run 5 times; the average of their recognition rates can be found in Table 4. The same set of training and test data was used for each run on each algorithm to better analyze the recognition rate. First off, it should be noted that none of the optimization algorithms attained results as accurate as those attained from the backpropagation method. This is not unexpected since backpropagation is known to work well in pattern recognition problems.[17] One reason why weights produced using the optimization algorithms may not work is that the weights produced through them likely suffer from overfitting; the weights are produced while minimizing the SSD without any cross-validation. Weights developed from a high dimensional function, which these have been, likely do not adapt well for real-life data since varying a parameter in such a function may have an overly strong effect. The regularization term, although it prevents blow-up, may also be preventing the best weights from being obtained. Of the algorithms tested, GA outperformed the other algorithms, attaining the highest accuracy to a notable degree, has the lowest number of evaluations, and an intermediate clock time. Its use of mutation and cross-over likely gave it the advantage over the other algorithms since it

[17] http://www.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html?searchHighlight=neural%20networks%20backpropagation%20speed%20and%20memory%20comparison

was not constrained to neighboring points. RHC had the second best accuracy, likely due to its large search area due to its polling function, but it also had the longest clock time and the most evaluations. SA produced the lowest accuracy results. This may be because its search range was constrained to randomly generated points from the data and accepted some values that did not optimize the function. This method may be inefficient for pattern recognition purposes or may be most susceptible to overfitting to the random points that were evaluated.

Table 4: Neural network training results

Algorithm	Clock Time	Func. Eval.'s	Recog. Rate, %
RHC - first poll	1301	5028	77
GA	923	3398	86
SA	521	3936	71

4 Conclusions

A summary evaluating the outcomes of the optimizing algorithms can be seen in Table 5. From these results, it appears that simulated annealing works well in real-life problems that are discreet function based rather than pattern based. This is because it is equipped such that it covers a large search area, it will not fall into local minima (as in the Rastrigin function), and searches points independent of starting point (as in the TSP problem). I believe it can be concluded that MIMIC works best when optimizing a discreet function that acts somewhat like a linear function such that it can make use of its ability to learn from previous iterations such that will interpolate previously evaluated points (as in the one-max problem). Of the optimization algorithms, the genetic algorithm did best when it came to pattern recognizing to produce weights for test data. Its use of mutation and cross-over likely gave it the advantage over the other algorithms since it was not constrained to neighboring points. However, it should be noted that it appears that backpropagation still seems to work best for pattern recognition problems, though this may be a result of overfitting when using the optimization algorithms due to the number of units and characteristics included in the function to be optimized.

Table 5: Overview comparing the performance of the optimizing algorithms for the test problems.

Parameter	Rastrigin	TSP	One-Max	Neural Network
Most Accurate	RHC/SA	RHC/SA	MIMIC	GA
Least Clock Time	SA	SA	RHC/GA	SA
Least Accurate	GA	GA	SA	SA
Most Clock Time	GA	RHC	MIMIC	RHC
Best Performer	**SA**	**SA**	**MIMIC**	**GA**