A Comparison of Randomized Optimization Methods

Chapman Siu

1 Optimization Problems

In this paper three optimization problems are chosen to demonstrate the various strengths of each algorithm, being four peaks, count ones and knapsack problem.

Four peaks and count ones were chosen due to their intuitive solutions which may not be obvious to machines. At the same time the representation of the two problems are very different; four peaks may lead an optimizer towards the local optima, whilst due to the oscillating effect in count ones, this does not occur.

The knapsack problem was chosen due to its ability assess a wide range of parameters and different scenarios which is helpful when comparing different optimization approaches. Furthermore since changes in configuration would lead to different solutions means that a greedy algorithm is unlikely to find the optimal knapsack solution.

1.1 Four Peaks

The Four Peaks problem is taken from (Baluja and Caruana, 1995). Given an N-dimensional input vector $\vec{X}$, the four peaks evaluation function is defined as:

$$f(\vec{X},T) = \max\left[tail(0,\vec{X}), head(1,\vec{X})\right] + R(\vec{X},T)$$

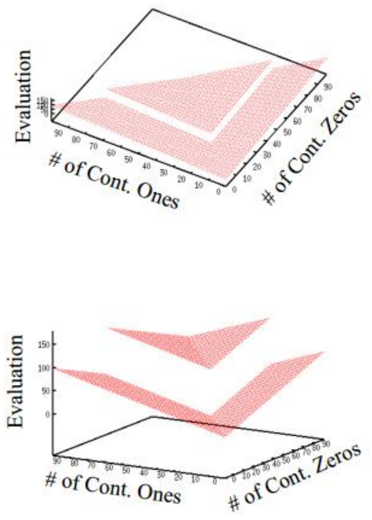
where

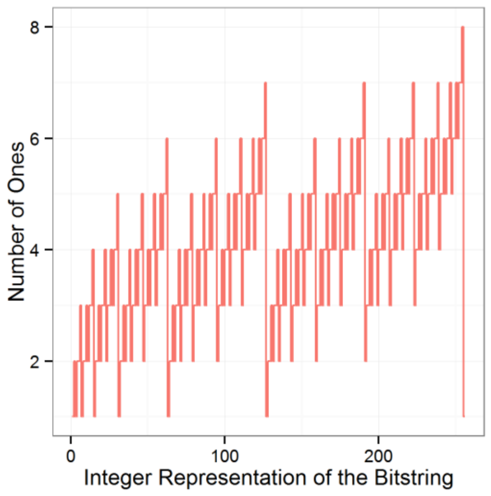
$$tail(0,\vec{X}) = \text{ number of trailing 0's in} \vec{X}$$

$$head(1,\vec{X}) = \text{ number of leading 1's in} \vec{X}$$

$$R(\vec{X},T) = \begin{cases} N & \text{if } tail(0,\vec{X}) > T \text{ and } head(1,\vec{X}) > T \\ 0 & \text{otherwise} \end{cases}$$

There are two global maxima for this function. They are achieved either when there are $T+1$ leading $1's$ followed by all 0's or when there are $T+1$ trailing 0's preceded by all 1's. There are also two suboptimal local maxima that occur with a string of all $1's$ or all 0's. Within this paper I have set $T = \frac{N}{5}$.





Left: Two views of the same four peaks problem, when N=100 and T=N/5 (Baluja and Caruana 1995).
Right: The value of count ones with respect to the integer representation of a bit-vector.

1.2 Count ones

Count ones is a discrete function defined for a bit-vector of length N. The goal is to search for an 1 filled solution. This can be defined by trying to maximize:

$$f(\boldsymbol{X}) = \sum_{i=0}^{n} g(x_i)$$

Where $g(x) = \begin{cases} 1 \text{ if } x = 1 \\ 0 \text{ otherwise} \end{cases}$.

Although this function is obvious to a human, the function oscillates with integer representation, since small changes in x (i.e. flipping a bit) will lead to small changes in the results as shown in the graph below.

1.3 Knapsack Problem

The knapsack problem is a constrained optimization problem: given a set of items, each with a mass and a value, determined the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

The knapsack problem can be formulated as follows. Let there be n items, z_1 to z_n where z_i has a non-negative value v_i and non-negative weight w_i. x_i is the number of copies of the item z_i. This is subject to the constraint W, which is the weight we can carry, and also c_i which is the total number of copies we have for each item x_i. Then we must attempt to maximize:

$$\sum_{i=1}^{n} v_i x_i \text{ subject to } \sum_{i=1}^{n} w_i x_i \le W, \quad x_i \in \{0, 1, \dots, c_i\}$$

2 Methodology

To solve the three problems above, the ABAGAIL library was used. This library has been modified, instructions on how to reproduce my results can be found in README.

For the neural network problem, MATLAB's implementation of genetic algorithm, simulated annealing and randomized hill climbing was used.

2.1 Parameter Tuning

Each dimension in the three problems above were run 10 times and the results averaged. The choice of parameters for SA, GA, and MIMIC were determined by an exhaustive grid search.

For SA the parameter varied was:

- Cooling Schedule - 0.7, 0.8, 0.95

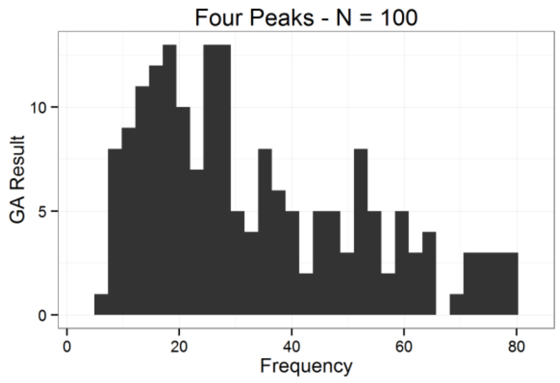
For GA, five parameters were varied:

- Population Size - 50%, 100%, 200%, 400% of Problem Size
- Crossover Type - One point, Two Point, Uniform
- Crossover Rate - 10%, 40%, 60%, 80%, 100%
- Mutation Rate - 0.001, 0.01, 0.1, 0.2, 0.5, 0.8, 1

For MIMIC, two parameters were varied:

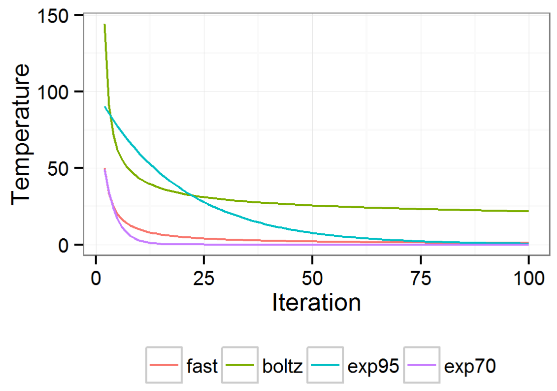
- Number of Samples taken each iteration - 40, 60, 80, 100, 120, 140
- Number of Samples to be kept each iteration - 20%, 50%, 70%, 90%

Parameter choice does make a difference to the performance of the particular algorithm. The graphs and table below demonstrate impact of choice for four peaks and determining weights for neural networks.



Problem Size	Population Size	Crossover Type	Crossover Rate	Mutation Rate
20	200%	Uniform	80%	10%
40	400%	Two Point	60%	20%
60	400%	One Point	100%	20%
80	400%	One Point	100%	20%
100	400%	One Point	60%	40%

Left: histogram of GA optimal result based on different choice of parameters, Right: Optimal Parameters selected by exhaustive grid search for GA



Cooling Function	MSE
Exponential: $T_k = T_0 \times 0.95^k$	0.1663
Exponential: $T_k = T_0 \times 0.8^k$	0.2080
Exponential: $T_k = T_0 \times 0.7^k$	0.2296
Boltzman: $T_k = T_0 / \log(k)$	0.2273
Fast: $T_k = T_0 / k$	0.2062

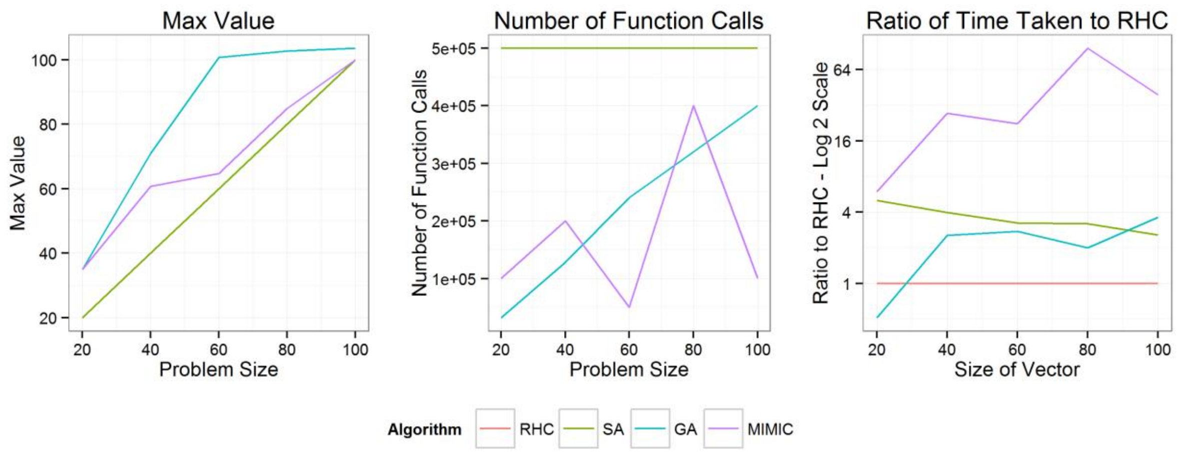
Left: Graph showing the temperatures of various SA cooling functions at various iterations, Right: the MSE of neural networks weights chosen by SA, where T_k is the temperature at time k.

The graph and table above demonstrates why choice of parameters is important. The histogram shows the spread of possible values for Four Peaks with problem size = 100 for GA only, this shows a wide range of values. The table above also shows the optimal parameters for the Four peaks for different problem size. It is clear that different problems will have different solutions and thus need to be treated differently. The choice of optimal parameter was performed for each of the three problems.

3 Results

3.1 Four Peaks

Four Peaks



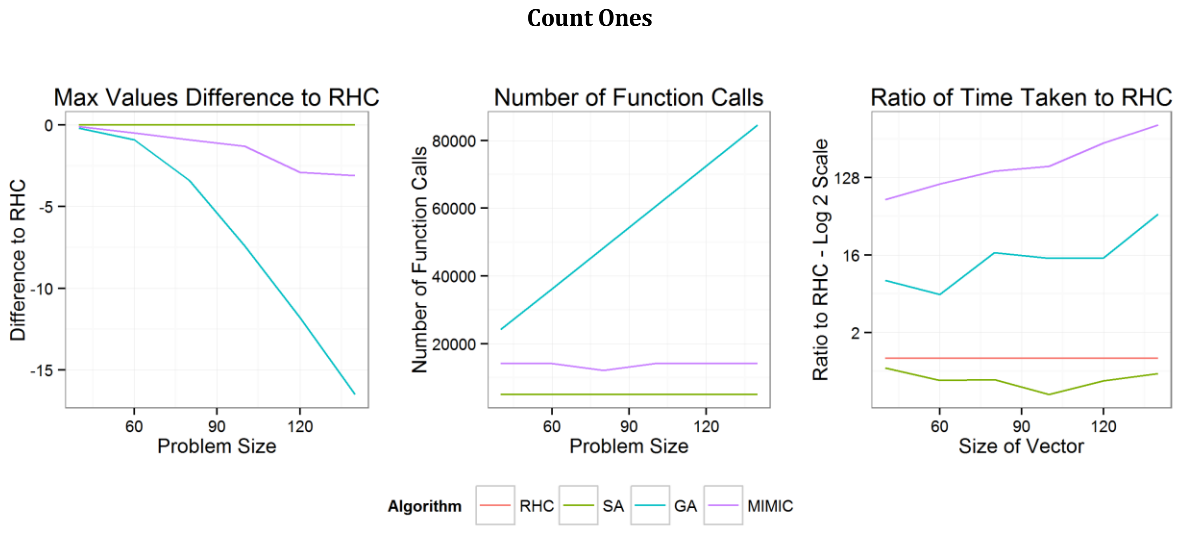
Four peaks was run for $n = \{20, 40, 60, 80, 100\}$, where n is the dimension of the vector $\vec{X}$ as defined in 1.1. The three graphs below show the average maximum value found by each algorithm for each problem size, The number of function calls for each algorithm and the time taken as a ratio to RHC.

GA is clearly the best algorithm, having the highest value, requiring the moderate number of function calls and very quick from a time perspective. In fact, on problem sizes 20, 40 and 60, GA indeed finds the global maximum.

Looking at the number of function evaluations completed by each algorithm, we MIMIC and GA have less evaluations that RHC and SA whilst generally achieving better scores, with both algorithms taking increasingly longer relative to the size of the problem size.

In comparison, RHC and SA only managed to reach the local maxima for each problem size. This is a reflection of the greedy nature of the algorithm, which fails to cross the large gap which is present in the four peaks problem as drawn in section 1.1. This highlights the greedy nature of SA and RHC, since it has become stuck on the local optima. In contrast, MIMIC's ability to discern structure and history has allowed it to perform better than the local optima, whilst GA's crossover operation has allowed it to propagate into the optimal solution.

3.2 Count ones Function



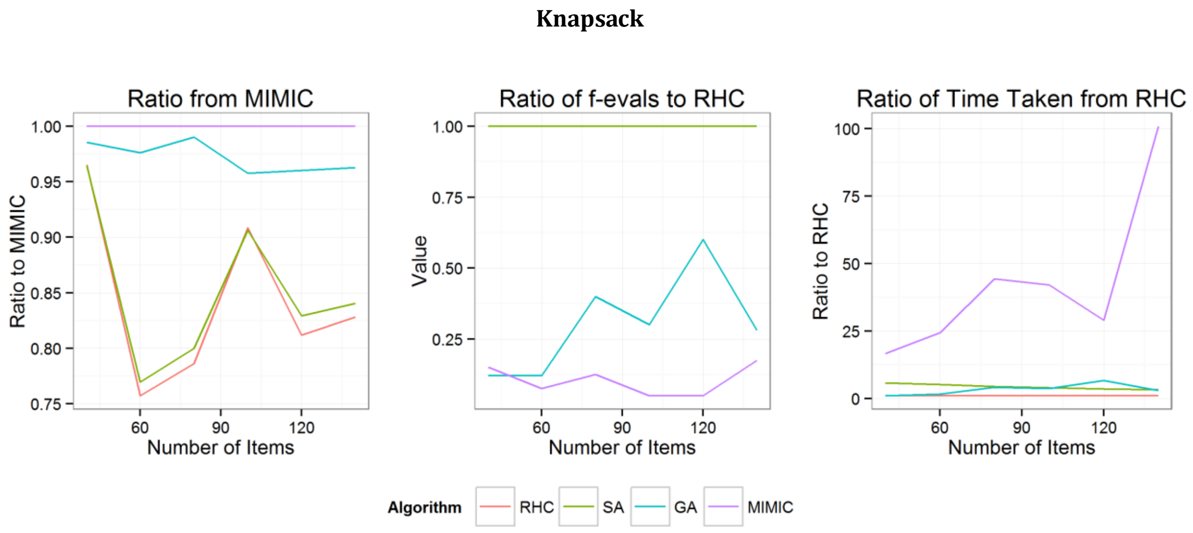
Count ones was run for $n = \{40, 60, 80, 100, 120, 140\}$. where n is the size of the problem as defined in 1.2. For every problem size, SA and RHC algorithms managed to reach the optimal point for this problem.

From the graphs above, SA and RHC actually reach the optimal value, with the absolute difference between MIMIC and the optimal result being minimal, whilst GA's optimal solution appears to be degrading at an exponential rate.

This problem reflects the performance of RHC and SA when there is no discernable structure to the problem, as shown in section 1.2. Since GA does not gain much from mutation, and MIMIC's knowledge of past points does not truly assist in improving the performance, this allows RHC and SA to excel. Furthermore SA and RHC only required 5,000 iterations to reach the optimal point, unlike MIMIC or GA which required over 10,000 iterations to reach a sub-optimal solution.

Count ones contrasts well with four peaks as it highlights situations where SA and RHC would perform well, compared with MIMIC and GA. When there is structure, GA and MIMIC's representation allow it to find better points, whilst when the problem at hand lacks such a structure, RHC and SA would perform better.

3.3 Knapsack



The knapsack problem was configured by varying the number of items, n, which were generated for $n = \{40, 60, 80, 100, 120, 140\}$, the maximum weight and maximum volume of the items were held constant at 50 for both, with weights and volume of each item randomized for each particular run.

The graph shows the relationship between MIMIC, the best performing algorithm in this example compared with the other 3 algorithms. It was completed this way since Knapsack does not have an analytical solution and the optimization problem is NP-hard. Overall it is clear that MIMIC and GA has the best performance, not only in terms of maximizing the value, but also in number of function evaluations, whilst RHC and SA have extremely similar results.

Knapsack highlights the greedy nature of RHC and SA, since from the top left graph we can see the erratic results that RHC and SA have compared with MIMIC and GA which have a consistent relationship. This is further evidenced by the fact that despite having twice as many function calls as MIMIC and GA, RHC and SA still perform at least 10% worse.

One reason why GA performs comparably to MIMIC and well compared with RHC and SA for the knapsack problem is due to GA's attempt to "capture structure by an ad hoc embedding of the parameters onto a line (the chromosome)" (De Bonet, Isbell, and Viola 1997). The ability to perform the crossover operation would be equivalent to sampling from a distribution, thus allowing GA to match the underlying structure of the problem. This allows GA to perform stronger than RHC and SA which does not utilize the structure in the optimization problem.

Thus in conclusion, for problems where retaining history would assist in creating a better optimization problem, MIMIC or GA would be a better performing algorithm, as indicated by Four Peaks and Knapsack problems. However, when the optimal point is random over a wide range of values RHC and SA may be the better solution as shown by Count ones.

4 Neural Network

The Pima Indian Diabetes data set from UCI Machine Learning repository was used for this portion of analysis.

The standard minimization of sum of squared error was used with regularization term as the function to minimize. Each algorithm was run until it met the tolerance of 1e-6 or 20 minutes of run time, whichever came first.

RHC was implemented using MATLAB function "patternsearch", whilst GA was implemented using MATLAB function "ga". SA was implemented through writing our own implementation, this may explain the lower number of function evaluations which SA had, despite having similar run time as the other two algorithms. The results were optimized and chosen based on the lowest validation error found and are tabulated below.

Algorithm	**Optimization Time**	**Function-Evaluations**	**Training Error**	**Validation Error**	**Testing Error**
RHC	20 min	33,674	0.1503	0.1663	0.1820
SA	20 min	20,275	0.1415	0.1739	0.1651
GA	20 min	30,021	0.1528	0.1755	0.1777
Backpropagation			0.1376	0.1652	0.1666

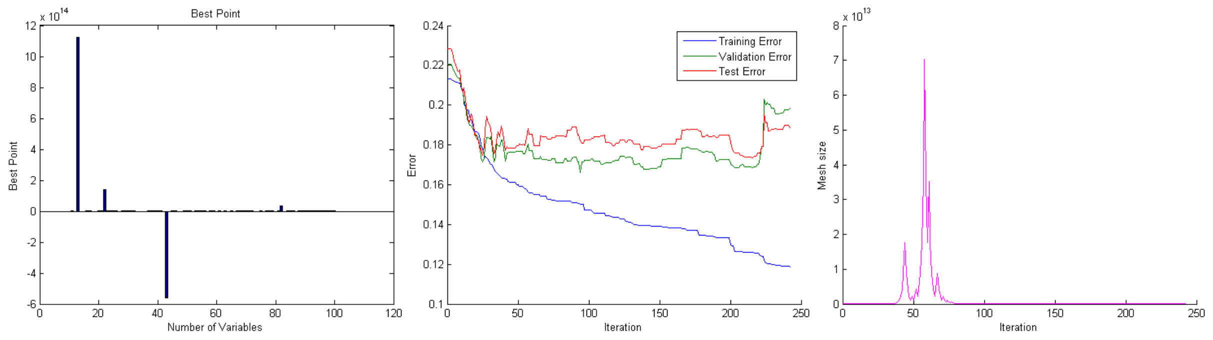
All approaches had very little to differentiate from each other, however we will examine how the inherent nature of the three randomized optimization algorithms result in different weights for the neural networks.

4.1 RHC

The graphs below show the progression of the search for the minimum point as well as the number of function evaluations at each iteration. Since this algorithm was allowed to run for 20 minutes, the algorithm did not terminate early when caught in a local minima, instead it expanded the search mesh in order to determine if any better points can be found in its vicinity. This is reflected at around iterations 50 and 70, where the mesh size peaked and the values on the best function value graph started to stall.

Again, RHC demonstrates its greediness when we examine the weights chosen. It has clearly simply chosen a few features which it then gave weightings in excess of 2×10^{14}

Left to right: The best point based on the minimizing validation error, the training, validation and test errors over RHC iterations, and the mesh size searched against



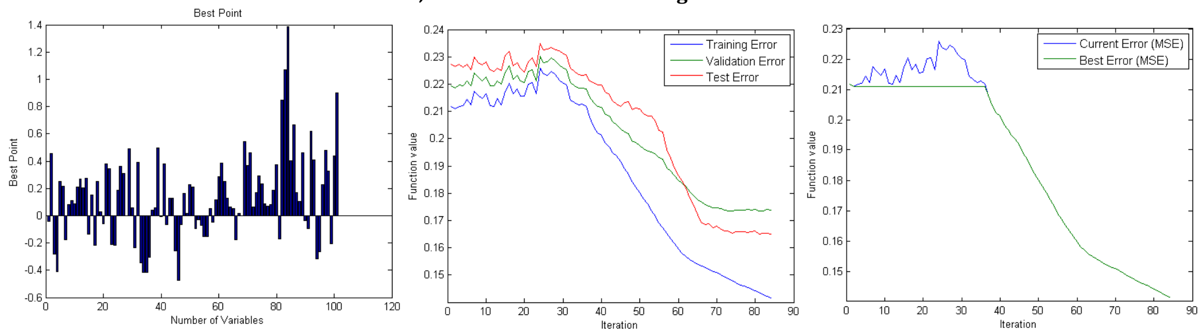
4.2 SA

MATLAB's native implementation was tested and but ultimately not used. This was due to the default settings of reannealing, which performed worse than a custom implementation with no reannealing. Similar to the three problems above, different cooling schedules (Exponential, Fast and Boltzman annealing schedule) and parameters, were tested and the best one was chosen and shown in the graphs below.

In contrast to the results from RHC, we can see that at least in the first 30 iterations, SA try to look around and accept points may not be the best point. However after the temperature falls down low enough it does indeed behave quite similar to RHC. From the number of function evaluations, we can see that it evaluated the loss function 40% less times than RHC, and have much fewer iterations. Despite this SA performed with similar MSE compared with RHC, which is a reflection of the hybrid nature of the algorithm; allowing for random walk-like behavior at higher temperatures and RHC behavior at lower temperatures.

This is further evidenced by the final point selected, which has a much larger mix of weights chosen for the hidden layer compared with RHC, with also a much smaller scaling.

Left to right: The best point based on the minimizing validation error, the training, validation and test errors over RHC iterations, and the route which SA algorithm travelled in.

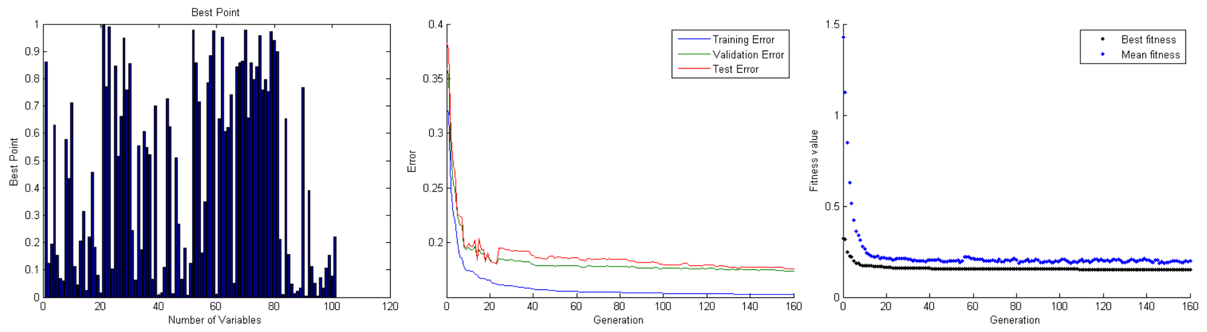


Judging by the validation and test plot, it appears that SA does not suffer from as much overfitting compared to RHC since it was much later that the validation and test error began to bottom out. This is confirmed in the tabular results above. This mirrors the temperature schedule (see graph in 2.1 for "exp95" to compare the temperature decrease rate with the training error graphs) which will begin to flatten out the errors when the algorithm begins to act more like RHC.

4.3 GA

Parameter tuning via exhaustive search was used to determine the optimal parameters as described in section 2.1. Judging solely by the graphs, it would appear with more lax conditions, GA would have terminated early by generation 40. However it continue to try to find better points, which we can see from the decrease in error between validation and test error in the later generations. This strong performance in GA is most likely related to the cross over model structural changes of connections of the layers and therefore lead to good results.

Left to right: The best point based on the minimizing validation error, the training, validation and test errors over RHC iterations, and the best and mean fitness of the population for each generation.



Again, from the best point, we can see how it differs from the greedy nature of RHC, since it has a much richer profile similar to SA.

If the best algorithm was based only on the numbers presented in the original table, it would be different to list a preference, however by examining these graphs, GA appears to be the strongest performing algorithm since under more lax conditions it would still appear to perform well.

5 Conclusion

From the experiments performed above, we can see the close relationship between SA and RHC. Both of these routines had extremely similar performance with SA requiring less function evaluations to get a similar performance to RHC.

For more complex problems where structure is important, MIMIC and GA performed the best. For exhaustive search problem SA or RHC would be preferable depending on the function evaluation cost.

Ultimately it depends on the nature of the problem and what particular factors are most important which will determine which of the four algorithms is the best algorithm to use. For a time sensitive problem, RHC or SA would be preferable, however when functions are expensive to compute, then MIMIC would be the best option.

6 References

Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. Technical report, Carnegie Mellon Univerisity.

De Bonet, JS., Isbell C, and Viola P (1997). MIMIC: Finding Optima by Estimating Probability Densities. Massachusetts Institute of Technology