Andrew Cudworth
CS XXXX Machine Learning (Spring 2020)
Assignment 2 Randomized Optimization

CS XXXX Assignment 2 Randomized Optimization

Four local random search methods including Random Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithms (GA) and Mutual Information Maximizing Input Clustering (MIMIC) were explored and compared via intentionally chosen optimization problems and training the weights on a Neural Network using a popular python library MLRose-hiive [1]. 3 optimization problems were defined as a state space and fitness function to be maximized. They included well known search benchmarks N-Queens, k-colors, and 4-Peaks discussed below. Performance was measured using fitness evaluation time, fitness per iteration and average iteration time among the optimizations and algorithms. A neural net previously trained on Airbnb data [7] was reproduced and trained using RHC, SA, and Gas and a log loss function.

PART 1 Optimization Problems

N-Queens

The N-Queens problem uses and NxN chess board and demands that N queens be placed on the chess board such that they do not intersect horizontally, Vertically or diagonally. An example of the solution is presented in Figure 1 for the N=8 Queens base case. The state space of the problem is represented as a 1-D array of length N where each index corresponds to a column on the board. For example, Figure 1 would be represented as [8,6,4,2,7,5,1]. The state space has size N^N. Fitness (or cost function) is evaluated by counting the number of queens intersecting and seeks to be minimized with global minima of 0. This calculation for the base 8 queens examples is fast at effectively zero in Table 2. However, fitness evaluation time scales linearly with the size of the problem as seen in Figure 4. Of the 3 optimization problems fitness evaluation time grows the most with input size for N-Queens. Importantly, there are multiple global minima distributed randomly throughout the state space.

K-Colors

The Max-K-colors optimization problem involves assigning a color to each node in a graph such that nodes sharing an edge do not share a color. The shape of the graph can take any form making the state space infinite. However, the problem is inspired by the travails of brave cartographers and coloring a map of Australia is a well explored problem [6 p.204]. Figure 2 shows a node representation of this map (excluding Tasmania for simplicity) which has 6 nodes and 9 edges that can be colored by a minimum of k=3 colors. [6]

For a single Australia, the state space is represented as 1D array of length 6 where each index is a node and colors take on numeric values between 0 and 2. The edges are defined separately as part of the problem itself. The fitness function (again a cost function) evaluates how may nodes sharing an edge have the same color. This can be minimized to 0 with run time 9.97E-4 on the base case. When scaling this problem for analysis the graph of Australia will be connected to a copy of itself via a single edge at Victoria. This maintains the minimum of 3 colors needed to color the graph. The state space grows by 3^{6*N} where N is the number of Australias. Evaluation time of a single fitness function on the base case is the longest on the optimization problems. Evaluation times scale linearly with input size at a lower rate than the N-Queens problem. Global and local minima are distributed randomly through the state space.

Andrew Cudworth
CS XXXX Machine Learning (Spring 2020)
Assignment 2 Randomized Optimization

4 Peaks

The 4 peaks problem explored in [3] creates an optimization problem with 2 unique global maxima and 2 unique local minima for any sized input state space. The state space has size 2^N and is defined as a vector of length N with each index having value 0 or 1. The fitness function to be maximized defined in [3 p. 8] and takes in a state vector and parameter T. It returns a local maxima for an input having all 0s or all 1s and a global maxima for a state with T+1 “1s” followed by all 0s or T+1 “0s” preceded by all 1s. Non-peak values are function of their nearness to a peak via the fitness function [3 p. 8] This can be seen in figure 3 which presents the entire state space of an N=7 and T=2 problem. The fitness requires effectively 0s on the base case and is effectively constant with input size.

Table 1 Base Unit Optimization Problems

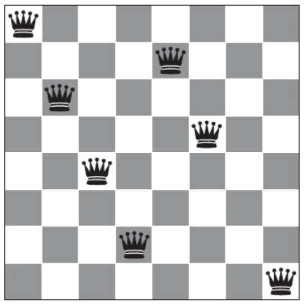
Figure 1 8 Queens Solution [8,6,4,2,7,5,1] [6, p. 123]

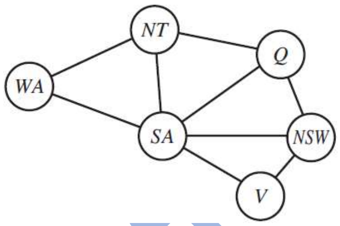

Figure 2 Three Color Graph Australia [6, p. 123]

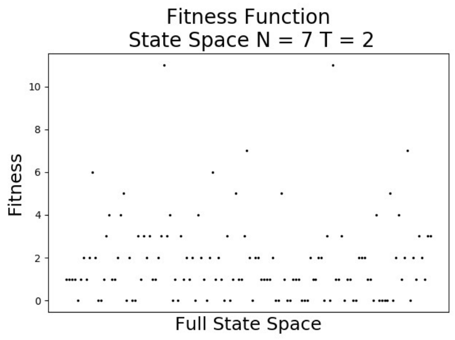

Figure 3 4 Peaks Fitness for N = 6

Results Summary

Fitness time vs input size was explored for each optimization problem. Plotted in Figure 4 and summarized in Table 2. N Queens grows most quickly followed by K-Color. 4 Peaks fitness evaluation time does not grow with input size.

Andrew Cudworth
CS XXXX Machine Learning (Spring 2020)
Assignment 2 Randomized Optimization

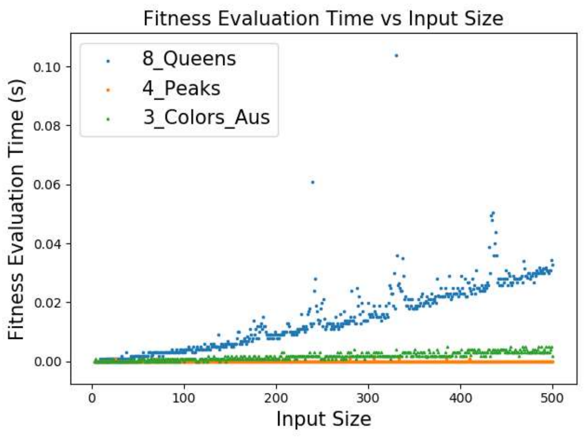

Figure 4 Fitness Evaluation Time

Table 2 Fitness Evaluation Time Growth

	Base Eval Time	Slope (s/Unit)	R2	Unit
N-Queens	~0.0	6.88E-05	0.88	"Queens"
K-Colors	9.97E-04	7.23E-06	0.87	"Australias"
4-Peaks	~0.0	~0.0	NA	"State Size"

To compare learning algorithms fitness was measured at each iteration of GA,RHC,SA, and MIMIC for the base case of each optimization problem. The fitness scores over all iterations were normalized and plotted in Figure 5 to allow comparisons. 4-Peak fitness scores were inverted and had the maximum shifted to 0. For K-Color and N-Queens, 0 is the best fitness (or cost) score. This means that for each optimization problem a score of 0 is the goal value. To account for randomness in the algorithms and optimization initial states, the process was repeated 10 times and the averages are plotted in Figure 5. The shaded area for each algorithm represents the standard deviation. Parameters for each algorithm are summarized in Table 3.

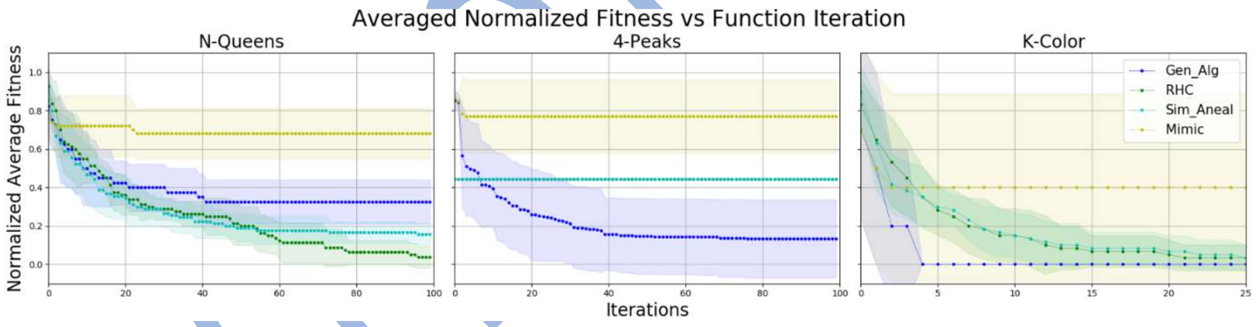

Figure 5 Mean Fitness vs Iterations

Table 3 Tuned Algorithm Parameters

Algorithm	Parameter	Value
RHC	restarts	20
SA	Decay Function	"Geometric"
SA	Init temp	1
SA	Decay Rate	0.5
SA	Min temp	0.001
GA	Population Size	20
GA	Breeding %	100%

Table 4 Iteration Average Wall Time (s)

Optimization	Algorithm	Avg. Iter Time (s)
N-Queens	MC	8.64E-2
N-Queens	RHC	1.10E-2
N-Queens	SA	1.60E-2
N-Queens	GA	6.15E-4
4-Peaks	MC	7.09E-1
4-Peaks	RHC	9.72E-4
4-Peaks	SA	5.69E-5

Andrew Cudworth
CS XXXX Machine Learning (Spring 2020)
Assignment 2 Randomized Optimization

MIMIC	Population Size	2000
MIMIC	Keep %	50%

4-Peaks	GA	3.18E-3
k-color	MC	4.25E-2
k-color	RHC	4.39E-5
k-color	SA	1.37E-3
k-color	GA	2.78E-2

For easier comparison, each algorithm was allowed as many attempts as needed per iteration to "increase" the fitness score. This means that the algorithm will only get better or remain constant each iteration in order to force convergence. Some weaknesses of this comparison is that iteration time varies for each algorithm as seen in Table 3 . For example, SA converges more quickly in terms of iterations for N-Queens. However, GA has the fastest performance per iteration. A more effective measure may be fitness evaluations taken to convergence. This was not explored due to implementation difficulties. Additionally, input size was held constant. However, complexity of the state space has an impact on algorithm performance.

Conclusions (Part 1)

Among 3 optimization problems, fit time as a function of input size scales linearly for N-Queens and k-Colors with N-Queens evaluation time increasing more quickly. Fit time for 4-Peaks is constant for input size. Normalized average fitness was compared across iterations designed to force improvement. For N-Queens RHC converged after the fewest iterations. However, GA had the shortest iteration and did no converge. MIMIC had the longest iteration time and worst performance. For 4-Peaks GA converged after the fewest iterations in some cases. However, its average fitness did not converge. However, GA had the 2nd longest iteration time after MIMIC. None of the other algorithms performed. For k-color GA converged after the fewest iterations. However, all algorithms performed well converging less than 25 iteration except for MIMIC. Iteration times were shorter for all algorithms as well with RHC having the shortest iteration time.

PART 2 Neural Net Weight Optimization

Neural Net weight training can be treated as an optimization problem where the weights are the state and the loss function is the fitness of the problem. This problem concept was applied to an existing NN on Airbnb data discussed in [7]. The 5x2x2x6 structure with 'relu' activation functions were maintained and the weight retrained via RHC, SA, and GA. A log loss function was tested to provide unique fitness results. Results are summarized in Table 5. Unfortunately, no optimization was found that did better than guessing the most frequent label. In fact, SA performed worse than most frequent guessing.

Table 5 Neural net Weight Optimization Results

	Train		Test	
Model	**Accuracy**	**Roc_Auc**	**Accuracy**	**Roc_Auc**
pre-trained	81.26%	63.6%	81.70%	63.5%
RHC	52.64%	50.19%	52.53%	50.25%
SA	46.40%	47.00%	45.70%	46.65%
GA	2.13%	50.00%	2.37%	50.00%

Andrew Cudworth
CS XXXX Machine Learning (Spring 2020)
Assignment 2 Randomized Optimization

SOURCES

[1] Hayes, G. (2019). *mlrose: Machine Learning, Randomized Optimization and SEarch package for Python*. https://github.com/gkhayes/mlrose.

[2] Hayes, G. (n.d.). Machine Learning, Randomized Optimization and SEarch¶. Retrieved from https://mlrose.readthedocs.io/en/stable/

[3] Isbell, C. L. (n.d.). Randomized Local Search as Successive Estimation of Probability Densities. Retrieved from https://www.cc.gatech.edu/~isbell/tutorials/mimic-tutorial.pdf03

[4] How to get all combination of n binary value? (2013, February 18). Retrieved from https://stackoverflow.com/questions/14931769/how-to-get-all-combination-of-n-binary-value

[5] Mitchell, T. M. (1997). Machine learning. New York: McGraw Hill.

[6] Russell, S. J., & Norvig, P. (2003). *Artificial intelligence: a modern approach by Stuart J. Russell and Peter Norvig; contributing writers, John F. Canny ...* Upper Saddle River, NJ: Prentice Hall/Pearson Education.

[7] acudworth3/Supervised_Learning_Exploration. (2020, February 23). Retrieved from https://github.com/acudworth3/Supervised_Learning_Exploration