Department of Computer Science, Georgia Institute of Technology

Randomized Optimization

Joseph Su

January 6, 2019

1 Introduction

This paper adapts and compares four local randomized search algorithms, Randomized Hill Climbing (RHC), Simulated Annealing (SA), Genetic Algorithm (GA), and Mutual-Information-Maximizing Input Clustering (MIMIC). We employ them to find optimized solutions in 4 problem domains:

- Neural Networks (NN) for Binary Classification
- Four Peaks, N-Queens, K-Colors

This paper is segmented into the above with their respective methodologies, evaluations and analysis. We treat our NN problem without back-propagation with RHC, SA, and GA, optimizing input weights to be continuous and real-valued. We then apply all of the algorithms to each of the three distinct optimization problems, Four Peaks, N Queens and K Colors, assessing each algorithm's weakness and strength.

Note the approach taken in this paper does not involve study of convergence, which largely depends on factors such as the problem domain and each algorithm's control parameters solving it. For example, it is hard to tell when an algorithm reaches convergence if a domain, such as Four Peaks, involves multiple local maxima. Instead, we run each algorithm through sufficient amount of iterations in the problem domains to assess their runtime and fitness based on empirical evidence.

Computing Facilities This work makes use of various libraries, toolsets, and modules in Python to arrive at its studies and results: `scikit-learn 1.19.dev0`, `numpy`, `scipy`, `IPython`, and `panda`. All of the algorithms are originally implemented in the open-source project ABAGAIL [2], and adapted for our experiments, which are performed on a 2015 Mac OSX 2.5 GHz Intel Core i7 with 16GB DDR3.

2 Neural Networks

The Wisconsin breast cancer dataset from Assignment 1 is chosen for this section. The dimension of this dataset is $X \in \mathbb{R}^{s \times f}$ where $s = 569$ samples and $f = 30$ features. The target label vector is $y \in \mathbb{R}^{f}$. The objective is to derive a hypothesis, $h \in H$, to approximate the target concept *c(x)*, where $x \in \mathbb{R}^{s}$. The optimal hypothesis, h', is one that minimizes the error function to map each $x \in X$ to either of the binary classification labels, benign (0) or malignant (1). We apply RHC, SA, and GA to optimize the weights in our neural networks.

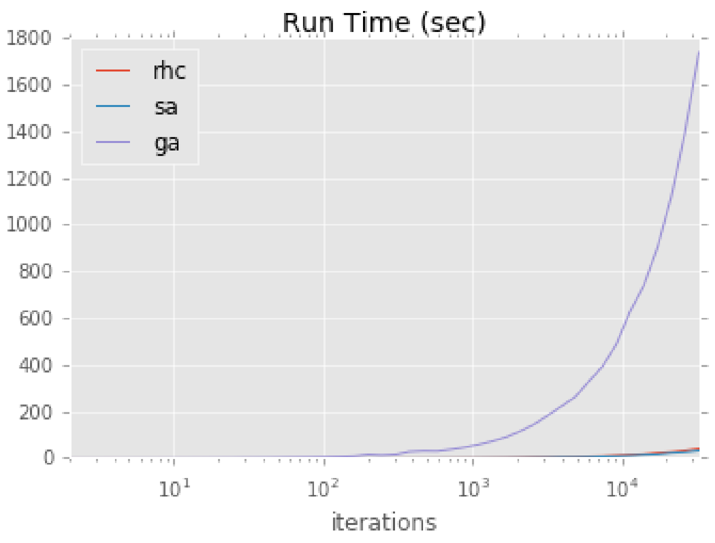

Figure 2.1: Runtime Comparisons on Neural Network

METHODOLOGY We employ `SumOfSquaresError` in ABAGAIL as our cost function, $SSE(h,s)$, where $s = |S|$ and S is our dataset. Our optimization objective is given by

$$SSE(h,s) = \operatorname*{argmin}_{h \in H} \left[\frac{1}{s} \sum_{i=1}^{s} (y_i^h - y_i)^2 \right] \tag{2.1}$$

We want to maximize our fitness objective, $SSE(h,s)^{-1}$. Our problem representation is adapted from `AbaloneTest` in ABAGAIL, with an `inputLayer` = 30 features and a softmax `outputLayer` = 1 for our binary classification. `LogisticSigmoid` as the activation function is used to allow for a continuous regression analysis of the problem. An `hiddenLayer` of 25 is chosen based on our grid search result from Assignment 1; this number produces the least amount of training and validation losses. A grid search is performed over GA's parameters but given its polynomial time we end up with a population size of 200, a mating rate of 100 and a mutation rate of 10. Similar search is performed over SA:

- Initial temperatures: 10^{11}, 10^{10}, and 10^{9}
- Cooling rates: 0.95, 0.7, 0.4, 0.1

By default SA has an initial temperature of $T = 10^{11}$ and a cooling rate of 0.95. We pre-process this dataset as we did in Assignment 1, by splitting the data into 80% training, and 20% held out, which are uniformly scaled down to zero mean and unit variance with the `StandardScaler` module in Python. The split yields 455 training samples, with 292 benigns (0) and 163 malignants (1). The holdout has 114 samples, with 77 benigns (0) and 37 malignants (1).

RESULTS We evaluate RHC, SA, and GA with respect to their predicability, time, and consistency on both training and holdout sets. The results are shown in SSE vs iterations and Runtime vs iterations in Fig. 2.1 and Fig. 2.2.

For the training set, as we increase iterations to 10,000, both RHC and SA start to show signs of overfitting beyond 5,000 (see Fig. 2.2). GA shows more resiliency than the other two algorithms as errors appear to have leveled off at the end; no overfitting is observed. This is not without cost: GA runtime (in seconds) increases exponentially after 800 iterations, as shown in Fig. 2.1. A rather pronounced fluctuation between iterations 100 through 500 is visible across all algorithms, especially for SA (Fig. 2.2). By lowering SA's initial T and speeding up its cooling rate the learning curve becomes more smooth and reaches its optimal fitness quicker albeit it diverges after 5,000 iterations as overfitting is inevitable. To see this, the confusion matrices show that for 500 iterations, SA is the worst candidate classifying many inputs to false positives, whereas both RHC and

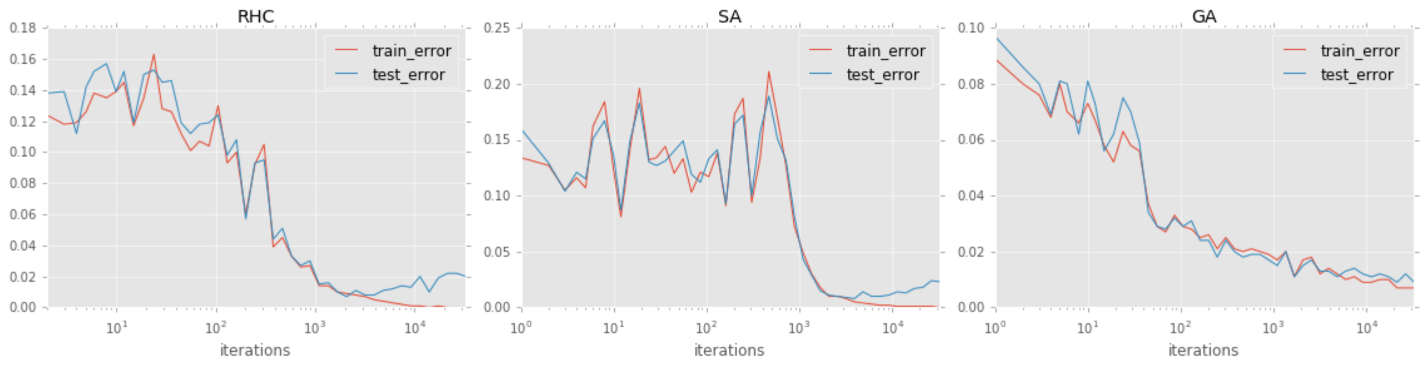

Figure 2.2: SSE Comparisons Over 10,000 Iterations on NN Training Set

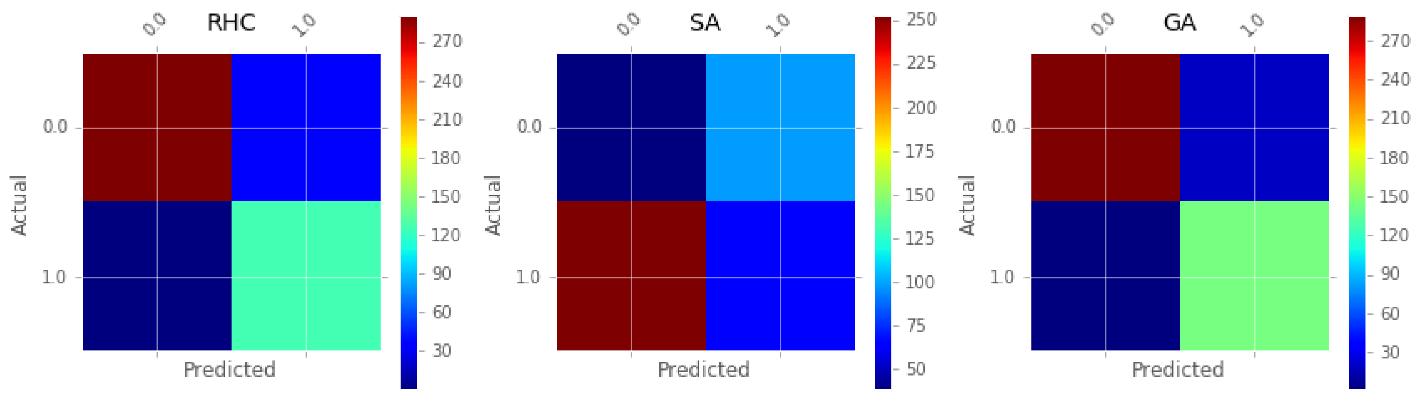

Figure 2.3: Training: RO Comparisons Over 500 Iterations on NN

GA assume better predicability, with RHC having more false negatives than GA. In diagnosing cancer cells we need to be extremely careful with RHC at low iterations to minimize false negative occurrence; in a clinical setting misclassification can lead to grave consequences.

For the testing set of 114 samples, we perform the same experiments as we did for the training. At 500 iterations, Fig. 2.4 bears similar characteristics as Fig. 2.3 in that SA misclassifies more inputs than either RHC or GA. Worst, SA has tendency to generate more false negatives than the other algorithms combined. Lastly RHC is less consistent than GA as the former produces more false negatives.

It is clear that the NN problem domain is not convex for SA with its chosen parameters. As the temperature cools, SA reaches an iteration threshold of 800-900 where SSE begins its rapid descend signaling the commencement of exploration over exploitation. With a slightly faster cooling rate, SA reaches that threshold sooner leading to hill climbing to its optimum. The problem is SA overfits as it reaches 5,000 iterations, just like RHC.

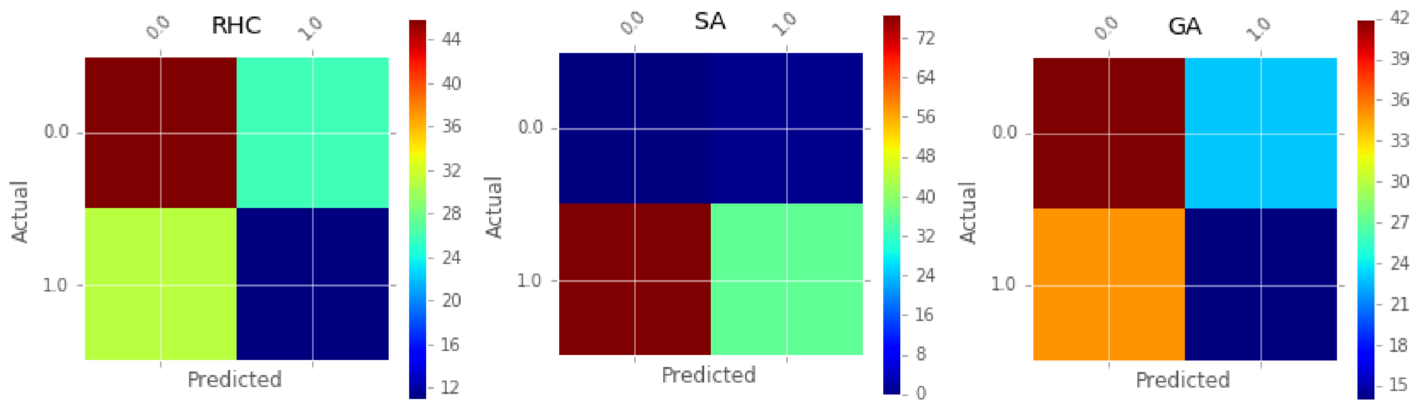

Figure 2.4: Testing: RO Comparisons Over 500 Iterations on NN

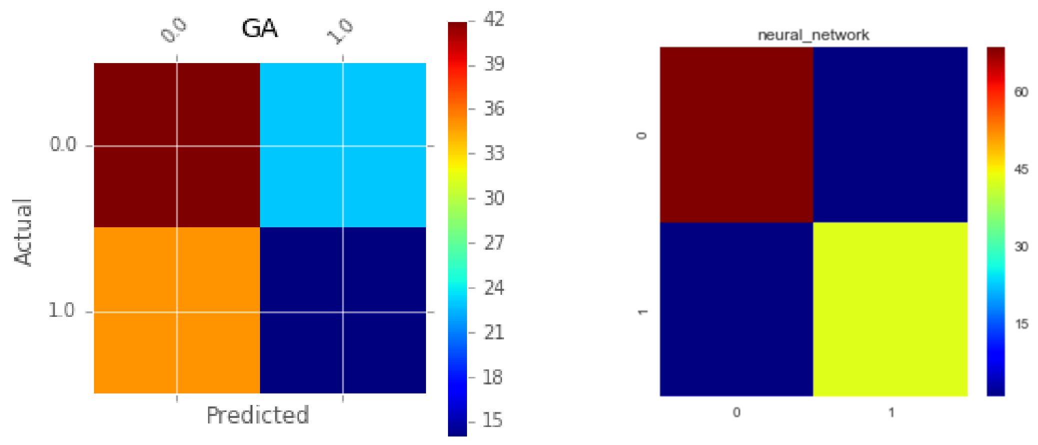

Figure 2.5: GA vs. Back-propagation (Assignment 1) over 144 Testing Samples

Fig. 2.5 shows the power of predicability between GA in this paper versus back-propagation in Assignment 1, over an holdout of 144 with sufficient iterations. The back-propagation technique employs a momentum rate of 0.9 and learning rate 0.01 over 200 epochs. Both GA and back-propagation correctly label (0,0) benign cells, however GA misclassifies many true malignant cells into either false positives or false negatives. Note both GA and back-propagation experiments have exactly the same number of input layers (30), hidden layers (25), and output layer (1) with binary classification.

3 Optimization Problems

We apply the 4 algorithms, RHC, SA, GA, and MIMIC, to 3 optimization problems: Four Peaks, N Queens, and K Colors. Henceforth we standardize our experiment setup to include the following:

- Input size, N, is a vector consisting of logarithmically spaced, positive, integral values on base 2.
- For each $n \in N$, we employ `FixedIterationTrainer` in ABAGAIL to iterate and ascertain an average fitness over n.
- Unless otherwise specified we run each algorithm with 1,000 iterations.

Note that fitness represents different measures for different problems. For Four Peaks and N Queens, a fitness score represents the optimal value over all iterations, whereas for K Coloring the same score represents the least number of iterations required to reach a coloring goal. Lastly, to ensure fairness and consistency, and to minimize idiosyncrasies associated with the computing environment, we run each problem 5 times to take the best average scores.

3.1 Four Peaks

This is a classic problem from [1]. Given n bits in an input vector $\overrightarrow{X}$ where $n \in \overrightarrow{X}$, and a difficulty parameter $T = N/5$, we have a four peaks fitness function

$$f(T, \overrightarrow{X}) = \texttt{MAX}[head(1, \overrightarrow{X}), tail(0, \overrightarrow{X})] + \texttt{Reward}(T, \overrightarrow{X}) \tag{3.1}$$

where $head(1, \overrightarrow{X})$ is the number of leading 1's in the vector, and $tail(0, \overrightarrow{X})$ is the number of trailing 0's in the same. $\texttt{Reward}(T, \overrightarrow{X}) = n$ if $(head(1, \overrightarrow{X}) > T) \wedge (tail(0, \overrightarrow{X}) > T)$, or 0 otherwise. There are 4 maxima, two global and two sub-optimal local, hence the name Four Peaks. Each of the 2 global maxima corresponds to a number with either more than $(T+1)$ leading 1's followed by 0's or more than $(T+1)$ trailing 0's preceded by 1's. The other 2 sub-optimal maxima correspond to when $head(1, \overrightarrow{X}) = n$ or $tail(0, \overrightarrow{X}) = n$ respectively.

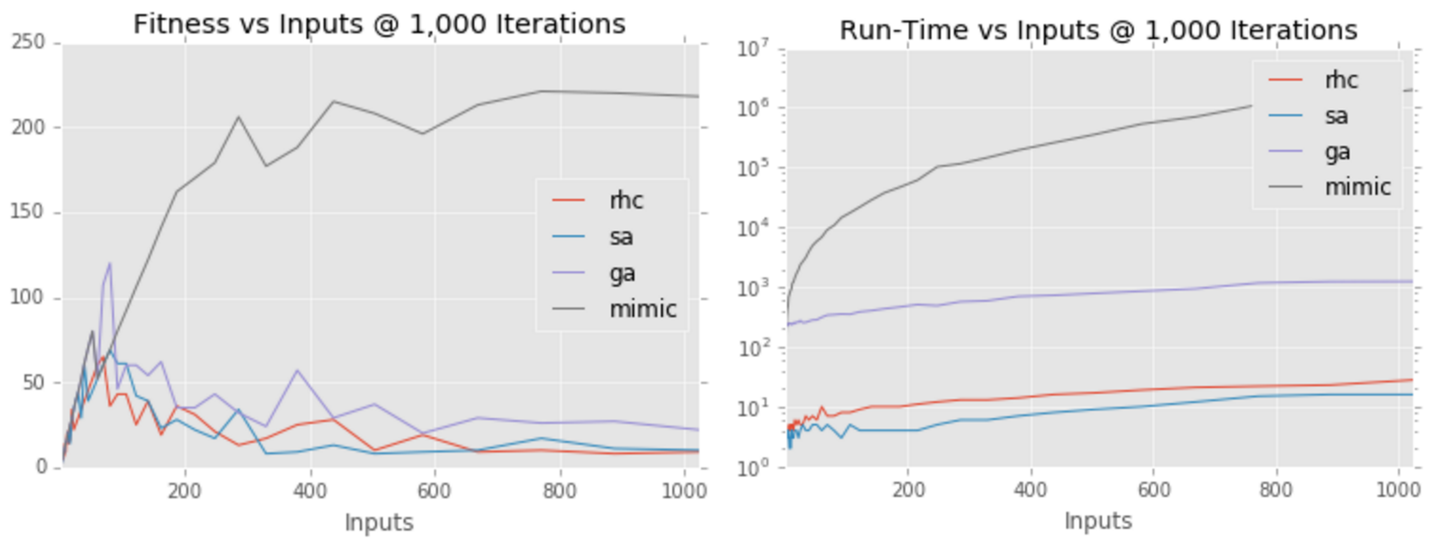

Figure 3.1: Four Peaks Fitness and Runtime Characteristics

METHODOLOGY In this experiment T is initialized to 20% of n, where n is logarithmically spaced $\in [1, 1024]$. We apply ABAGAIL's `FourPeaksEvaluationFunction` encapsulating the aforementioned algorithm. We run 1,000 iterations per input $x \in X$. The default `FourPeaksEvaluationFunction` is used. The setup for each algorithm is as follows:

- SA: $T = 10^{11}$, cooling rate = 0.95.
- GA: ABAGAIL's `SingleCrossOver` and `SwapMutation` are employed. Additionally, population = 200, mate=100, and mutate=10.
- MIMIC: sampling population = 200, cutoff threshold = 20.

RESULTS Fig 3.1 shows Fitness vs Runtime with respect to our Input space. All algorithms grow fitness linearly at $n < 20$, but with the exception of MIMIC, they start to flatten out as n increases to > 60. This trend is observed in Fig 3.2. The optimal fitness in Four Peaks with a sample size of 100 is 189 [1]. That fitness is achieved with a very large number of iterations. Our experiment has 1,000 iterations, leading to a score of 89 for MIMIC, 52 for GA and around 65 for SA and 45 for RHC, as seen in Fig. 3.2. This insinuates that all of the algorithms, at 1,000 iterations, fail to climb passed the steep local maxima located at either $head(1, \overrightarrow{X}) = 100$ or $tail(0, \overrightarrow{X}) = 100$, with RHC and SA getting stuck most of the time, unable to get out of the two maximum traps regardless how many bits are added to the shorter of the two ends. SA, on the other hand, improves slightly with a higher fitness overall when it starts the climb with a faster cooling rate and lower initial temperature, effectively starting its early marshall into exploitation, deepening its search into the local area with higher fitness. RHC is the worst of the bunch, unable to move to equally performing states by giving up its climbs making its reaching either global maximum highly improbable.

MIMIC and GA achieve much higher fitness than RHC and SA, insinuating their resiliency to climb passed the local maxima despite never having set foot in a new territory. MIMIC in particular surveys the entire landscape looking for large peaks, and repeatedly making the best decisions over time.

3.2 N QUEENS

The N-queens puzzle is the problem of placing N queens on an $N \times N$ chessboard such that no two queens attack each other. Note the solution for $N = 1$ is 1. This constraint problem has been extensively studied [3], and the algorithm for finding one solution shows relevancy to storage schemes, VLSI testing, deadlock prevention, traffic control, and other domains [4].

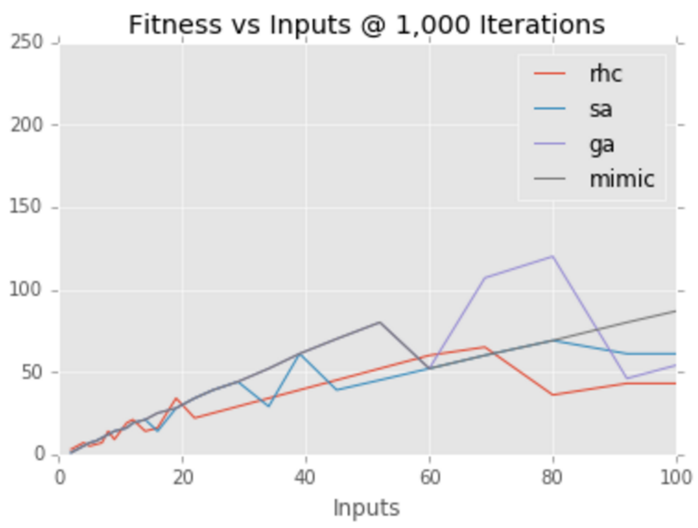

Figure 3.2: Four Peaks Fitness Closeup

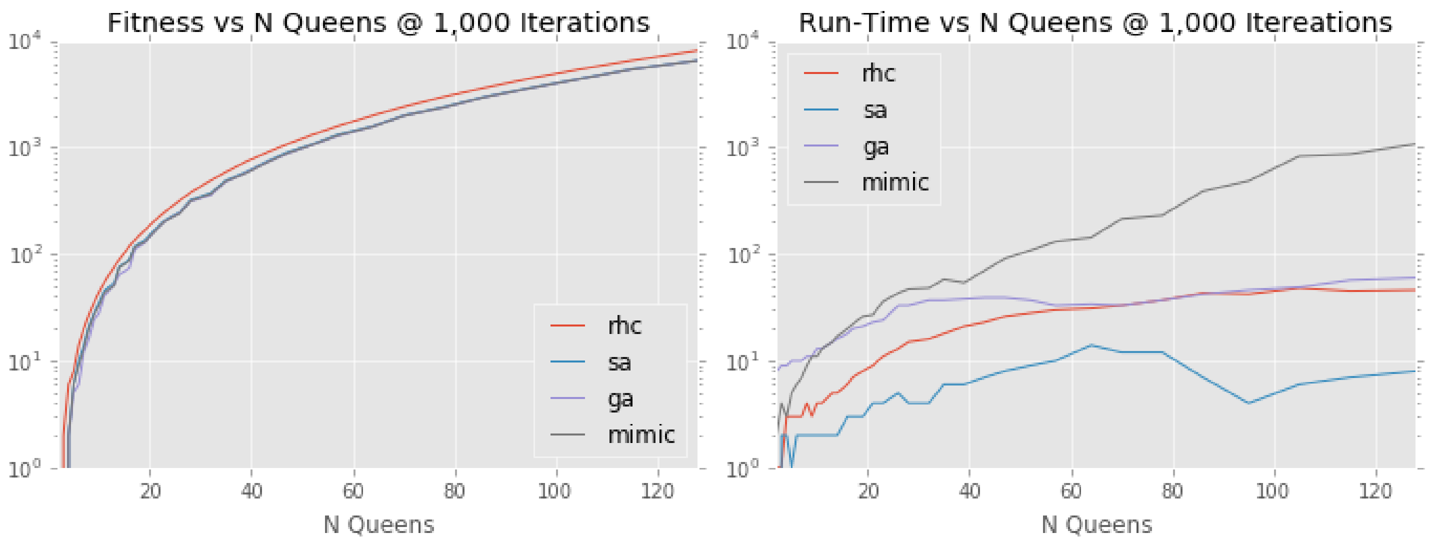

Figure 3.3: *N* Queens Fitness and Runtime Characteristics

METHODOLOGY We employ ABAGAIL's `NQueensFitnessFunction` to evaluate the number of steps required to find the first solution, namely the first board position with non-attacking pairs of queens. We have n number of queens on a chessboard, with n being logarithmically spaced $\in [1, 128]$. For each n we run the experiment a fixed number of times to reduce bias of any individual run:

- RHC, SA: 100 iterations are conducted with `FixedIterationTrainer` in ABAGAIL. For SA, we have $T = 10^{11}$ and cooling rate = 0.95 in one run, and the opposite extreme of $T = 10^1$ and cooling rate = 0.1 in another.
- GA: same exact setting as Four Peaks, with `SingleCrossOver` and population=200, mates=100, and mutations=10.
- MIMIC: sampling population = 200, cutoff threshold = 20. A total of 5 iterations conducted per input.

RESULTS The most indicative factor representing performance in this experiment is Runtime vs N queens. Varying control parameters of SA produces little effect on its performance characteristics. All algorithms appear to have achieved the same fitness trends as N increases in O(log n) time, which is not as surprising as

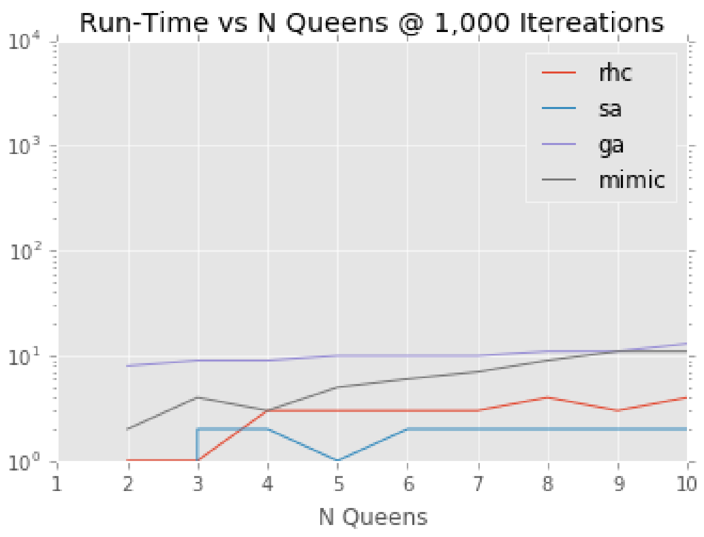

Figure 3.4: *N* Queens: Run Time Closeup

`NQueensFitnessFunction`'s objective function, despite its $O(n^2)$ with breadth-first traversal, has an underlying representation that leads to an uniformed set of function evaluations regardless of the size of *N* (see Table 1). Both RHC and SA show nearly O(1) which is clearly the case in Fig. 3.3 and Fig. 3.4.

algorithm	N-queens	fn_evals
RHC	4	101
SA	4	101
GA	4	1182
MIMIC	4	1200
...		
RHC	128	101
SA	128	101
GA	128	1176
MIMIC	128	1200

Table 1: Function evaluations vs algorithms vs N-queens

Given that SA consistently takes 101 function evaluations per *N* to reach the first solution, and that its combination of hill climbing and random walk lead to a very fast convergence to optimal fitness, SA is the winner in this experiment on both time and fitness induction. MIMIC has the worst performance; its time is on the order of $O(\log N)$ - indicative of the algorithm's inherent strength in divide-and-conquer, requiring exploration of only a fraction of the space despite the algorithm's need to construct minimum spanning trees and space search in depth.

3.3 *K* Colors

This problem considers a graph which is *K* colorable if we may assign *K* colors to each of the nodes such that no adjacent nodes have the same color.

Methodology We use `MaxKColorFitnessFunction` in ABAGAIL to evaluate the required iterations to arrive at a target coloring assignment given *K*. We set the number of vertices to 50 and adjacent nodes per vertex to 4. The coloring space, $\overrightarrow{K}$, is logarithmically spaced $\in [1, 128]$. We also have:

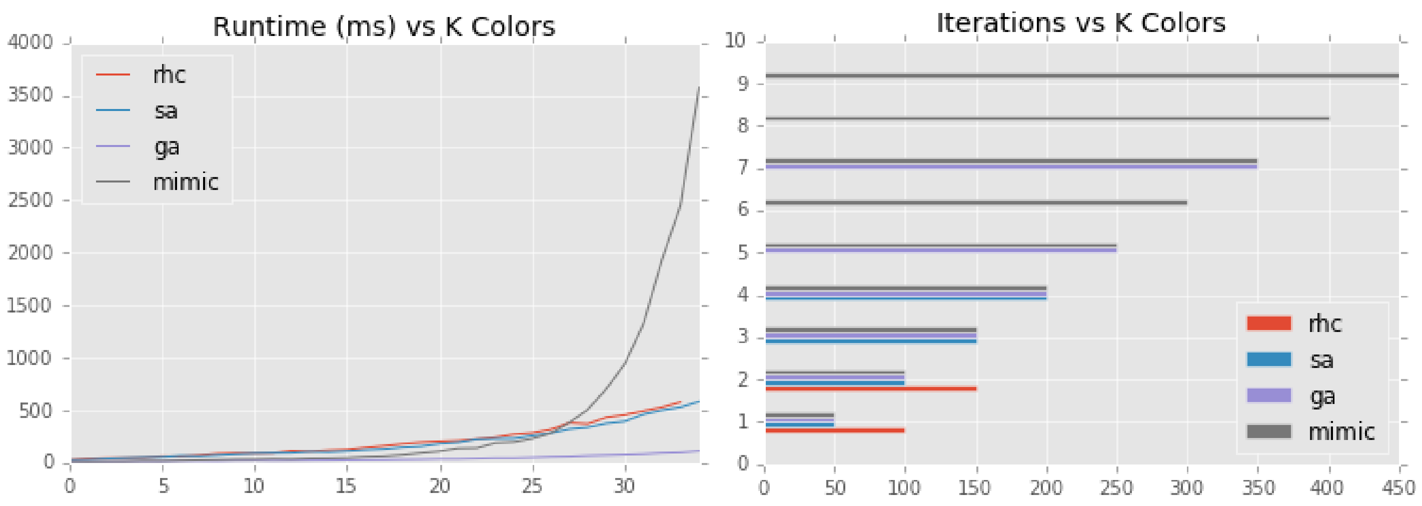

Figure 3.5: K Colors Iterations and Runtime Characteristics

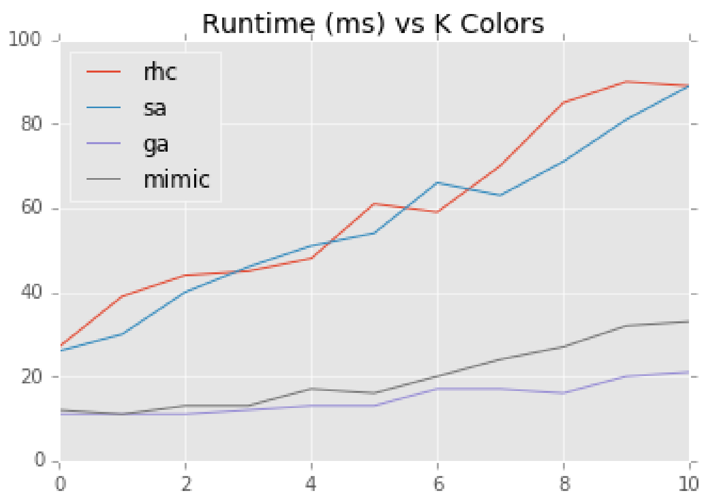

Figure 3.6: K Colors: Iterations for $K < 10$

- RHC, SA: 20,000 iterations are applied per population sample. For SA, we use $T = 10^{12}$ and a rapid cooling rate of 0.1.
- GA: default `SingleCrossOver` and `SwapMutation` in ABAGAIL are employed. For each iteration we use population=200, mates=10, and mutates=60. 50 iterations are conducted per population sample.
- MIMIC: sample population=200, cutoff=100. A total of 5 iterations is conducted per population sample.

RESULTS MIMIC leads to the best fitness (least amount of iterations required) as K increases. GA follows suit, with its ability to solve $K = 5$ and $K = 7$, whereas both RHC and SA stop short at $K = 4$. This behavior is observed in Fig. 3.5. Time wise, MIMIC requires polynomial time which becomes exponential when $K > 30$, making MIMIC the least viable option at higher order K. For $K < 10$, as seen in Fig. 3.6, one observes that both RHC and SA require $O(K)$ time whereas both MIMIC and GA are almost $O(1)$. In addition MIMIC is the only algorithm able to find solutions for $K > 6$ which indicates its advantage over other algorithms considered.

Both MIMIC and GA have representational advantage over others for K Colors, which may have multiple local maxima making this domain less representative for local-gradient techniques such as RHC and SA.

4 Discussion: Algorithms

We discuss and summarize the efficacies of our randomized optimizers applying to our domain problems.

RHC This is a greedy algorithm that updates its weight vector in the direction of the steepest hill. RHC lends an intuitive insight into the nature of the NN problem involving Wisconsin cancer dataset, with 30 features and < 500 rows. RHC appears to be a good choice for this binary classification problem with a unit space complexity O(1) and a linear time complexity O(n) with respect to the number of its neighboring states.

SA This is a gradient-based technique combining both hill climbing and random walk that yield both efficiency and completeness. SA is a good choice for N Queens, with its state of representation favoring making decisions stochastically to a global optimum.

GA This algorithm shows its strength for NN, Four Peaks and, generally, for problems involving black box search processes with ill-defined objective functions. Secondly, GA demonstrates that crossover and mutations in a population can lead to "fitter" individuals. This is illustrated in Four Peaks, where newly recombined candidates via single-point crossover are rewarded with a value of n for satisfying the $head(1, \overrightarrow{X}) > T \wedge tail(0, \overrightarrow{X}) > T$ constraints. GA requires carefully engineered state representation involving population size (n), crossover type/rate, mutation rate, etc. When tuned correctly for a problem without seemingly obvious objectives, GA may perform better than local gradient-based RHC and SA. Lastly, GA exhibits a space complexity of $O(n)$ and polynomial time complexity as its evolution commences.

MIMIC MIMIC out-performs all others in K Colors, which is NP-complete [5]. This is not surprising as in general MIMIC favors NP-hard and NP-complete problems as it persist good representations of the underlying hypothesis space, capturing much of the structural regularity within the inputs.

5 Conclusion

This paper studies and explores the application of four randomized optimization techniques over 4 different problem domains, and evaluates each algorithm's performance characteristics relating to runtime and fitness. For NN without back-propagation, GA is a good randomized optimizer but with a polynomial runtime. RHC with its local gradient technique is also good, with its agility to arrive at a reasonably accurate classification rate. Still, back-propagation with online weight updates and momentum (from Assignment 1) outperforms the best of randomized optimizers in consistency and predicability on a binary classification problem involving dataset < 1,000 rows and < 30 features. SA is best suited for problems involving multiple maxima. With a correctly tuned temperature-dependent time schedule, the algorithm trades off between exploration and exploitation. For instance, SA works very well in N Queens but fares poorly in NN (as evidenced by its error fluctuations in Fig. 2.2). MIMIC shows tremendous promise in K Colors, and generally NP-complete and hard problems by statistically sampling distribution favoring optimality. In summary, we conclude the representational structure of a problem domain is the primary factor governing the efficiency and efficacy of an algorithm.

References

[1] Baluja, S. Caruana, R (1995) Removing the genetics from the standard genetic algorithm. *Technical Report, Carnegie Mellon University*

[2] ABAGAIL (2016). GitHub repository. *https://github.com/pushkar/ABAGAIL)*. Web. 10 Oct. 2016.

[3] Hoffman, L. Constructions for the Solution of the m Queens Problem. *Mathematics Magazine*. p. 66-72, 1969.

[4] Jordan, B. A survey of known results and research areas for n-queens. *Discrete Mathematics*. Volume 309, Issue 1, pp 1-31, 2009.

[5] Bonet, J., Isbell, C., and Viola, P. (1996) Advances in neural information processing systems. *Conference: Advances in Neural Information Processing Systems 9*. NIPS, Denver, CO, USA.