Enmao Diao CS4641
Project 2

Randomized Optimization Report

DATASET

MUSK (Version 2) Dataset. This dataset describes a set of 102 molecules of which 39 are judged by human experts to be musks and the remaining 63 molecules are judged to be non-musks. The goal is to learn to predict whether new molecules will be musks or non-musks. However, the features that describe these molecules depend upon the exact shape, or conformation, of the molecule. A single molecule can have many different shapes. All the conformations of the molecules were generated to produce 6,598 conformations. The conformation name attributes are removed to create a pure numeric dataset. This dataset therefore contains 6,598 instances and 166 attributes.

NEURAL NETWORK OPTIMIZATION

APPROACH

To compare three optimization algorithms, Randomized Hill Climbing (RHC), Simulated Annealing (SA), and Genetic Algorithm (GA), with Back Propagation (BP) used in Project 1, we establish three metrics two judge their performance. The first is to analyze the accuracy versus iteration, because we are interested in the speed of three different algorithms. The accuracy versus run time is also measured because a direct comparison between some algorithms may have an inexpensive update step but require a lot of iterations to converge, whereas others may have a costly update step but require fewer iterations to converge. Secondly, we analyze the computational complexity by measuring the iterations and run time versus data sizes and model complexity. Originally, we use 70% dataset as training set and 30% as testset. Now we split the training dataset into subsets and test the model with the test set. We vary the model complexity by changing the number of hidden layers of Neural Network. We are interested in the computational complexity because we want to show that how each algorithm scales with different data size and model complexity. Finally, a conclusion is drawn based on our analysis.

SPEED & EFFICIENCY

As you can see from Figure 2 and 3, we plot the test accuracy vs iterations and run time. We fix the parameter value for SA, GA in this case to have a direct view of speed and efficiency. Four different algorithms behave very differently on the musk dataset.

RHC converges at about 1000 iterations and its run time is 50s. Although it is an inexpensive optimization algorithm but it turns out to be good enough for this problem. RHC is very different from back propagation with gradient descent methods, which adjust all of the values in at each iteration according to the gradient of the hill. With hill climbing, any change that improves function is accepted, and the process continues until no change can be found to improve the value which is the locally optimal. In this way, RHC believes data completely because it will accept any possible improvement move and as long as there are often convex hill climbing, RHC can reach a global maximal. Clearly, this dataset is well formed and precise to the classification problem so that the simplest optimization problem can deal with it very well. The local maxima that RHC found has a relatively larger covering space compared with both lower and higher maximal. Because it does not take many trials for RHC to find a good start point and reach that maximal and it converges at that local maxima. Based on the SA performance, I would argue that RHC does not reach the global maximal in this case but it reaches a high local maxima that is very close to a higher maximal.

SA converges at about 1500 iterations and its run time is slightly larger than 50s. Its performance when it converges is very similar to RHC. However, with small number of iterations, it has very poor performance. This is because, we set a high temperature value so that we will be able to reach global maximal by stepping over the local maxima. In this case, it takes about 1500 iterations to cool down which means that the randomness from the high temperature does not guarantee a local maxima with small amount of iterations. We set the cooling rate to 0.95, the way to improve the performance is to lower cooling coefficient or lower the initial temperature so that it can reach a local maxima with small amount of iterations, because this problem seems to have a very high local maxima that is close to the global maximal. As the iteration grows, SA is finally able to reach a higher maximal at 5000 iterations. RHC however, is not able to reach that higher maximal as it stuck at the local maxima which it reaches at 1000 iterations. I would argue that, with lower initial temperature value, SA can perform exactly like RHC with small amount iterations. However, it might lose the capability to reach the higher maximal because the temperature can be cooled down and stuck at lower maximal like RHC. In this case, because the higher maximal is not dominantly superior to the lower maximal. It does not worth running 5000 iterations and only get tiny improvement. However, as the data size and model complexity grows, the gap between maximal will become larger as we will show later. It is a tradeoff between performance with small amount of iterations and the possibility of reaching higher maximal with large amount of iterations.

In order to have run time that is comparable with RHC and SA, we scales down the number of iterations of GA 10 times. So we run 500 iterations for GA when RHC and SA run 5000 iterations because GA has expensive step. In this case, GA reaches the local maxima at about 20 iterations and only spends 7 seconds. In this case, this is dominantly superior to RHC because it perform better in the sense of both iterations and run time. In order to show the impact of different parameters of GA, I vary the population, mutation and cross over size as you can see in the Table II. Population size gives a basic guarantee for the performance because it is very hard to find a local maxima with small population size. In this case, if we only evaluate 10 random weights, we will be likely to miss any maximal values. The only hope is to increase the amount of iterations to compensate the loss of weights coverage. With higher crossover value, we take the advantage of good restarts and their good properties would possibly direct us to a higher maximal. We are able to jump over some local maxima with higher covering space like the one that RHC detected which yields 85.202%. The Table shows that higher crossover value reaches a higher local maxima than RHC does. However, with higher mutation value, we prefer local search to crossover. This leads us to reach the local maxima with larger covering space or basin of attraction. The Table shows that we are attracted by the local maxima that RHC detected when we have higher mutation value. Fortunately, in this problem many local maxima are very close to each other, it is acceptable to have smaller crossover and mutation values but we cannot have a very small population size.

BP is the best algorithm in the sense of iterations and run time. As you can see from the two figures, BP converges to the optimal value at about 61 iterations and 2 seconds. It also happens to reach one highest maximal at 575 iterations and 20 seconds. I believe the reason is that the dataset is extremely well formed for classification problem. The model has a local maxima with large covering space and also yields good performance from that maximal. Without optimization, pure BP is good enough to deal with the problem because it can easily reach that optimal local maxima without additional cost. However, all optimization algorithms have some kinds of randomness which actually happens to yield worse performance. The model is so simple that BP can sufficiently deal with. There is also additional computational cost using optimization. Therefore, in this case, there is no need to use any sort of optimization algorithms on this dataset. In Table I, we summarize and rank the algorithms in the sense of iterations and run time. We will analyze the algorithms in details with more general problems like Four Peaks. In the next subsection, we will analyze the scalability of each algorithm with respect of data size and model complexity.

BP is clearly the best in this case because it reaches 85.202% at only 61 iterations and 2 seconds and it can also yield the highest accuracy when given more time. GA ranks the second because it is superior to RHC on all three metrics as shown in Table I. I rank RHC and SA the same because I believe there could be a trade-off between the best performance and run time for these two algorithms. In this case, RHC can reach the optimal local maxima with constant time but it does not yield accuracy higher than that. SA retains the possibility of reaching higher local maxima. Although it may take a long time to reach the higher local maxima, if we tuning the parameters properly, it may manage to do it within a small amount of time. For problems where finding the precise global maximal is less important than finding an acceptable local maxima in a fixed amount of time, SA may be preferable.

Table I
Rank for optimization algorithms on musk dataset

Algorithms	Rank	Best performance	iteration	run time (s)
BP	1	88.687	575	22.881
GA	2	85.202	20	7.758
RHC	3	85.202	1000	37.778
SA	3	87.879	5000	87.879

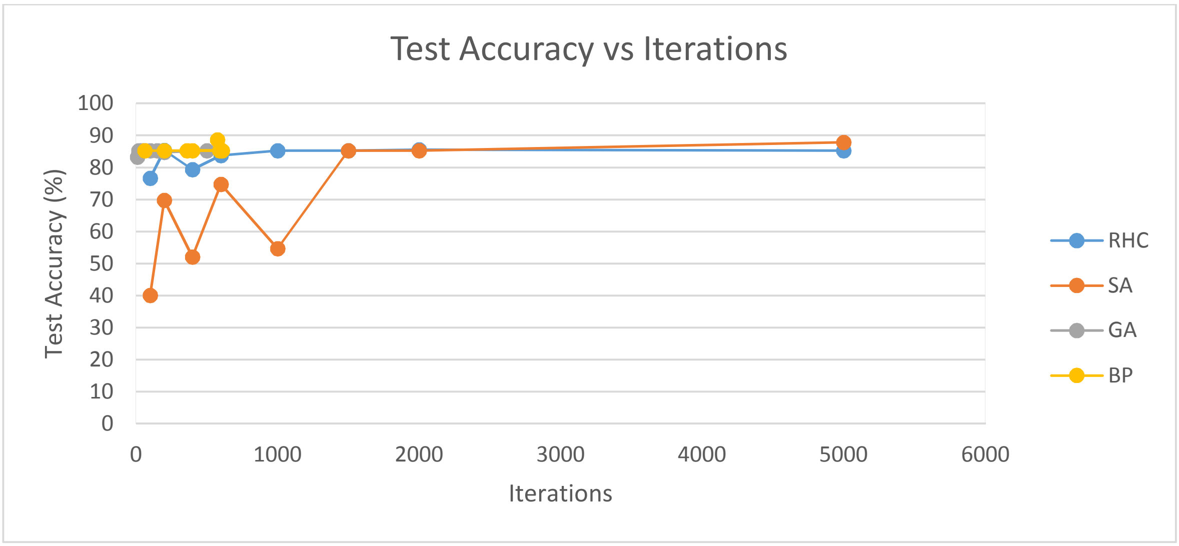

Figure 2. Test accuracy vs Iterations for four different algorithms.

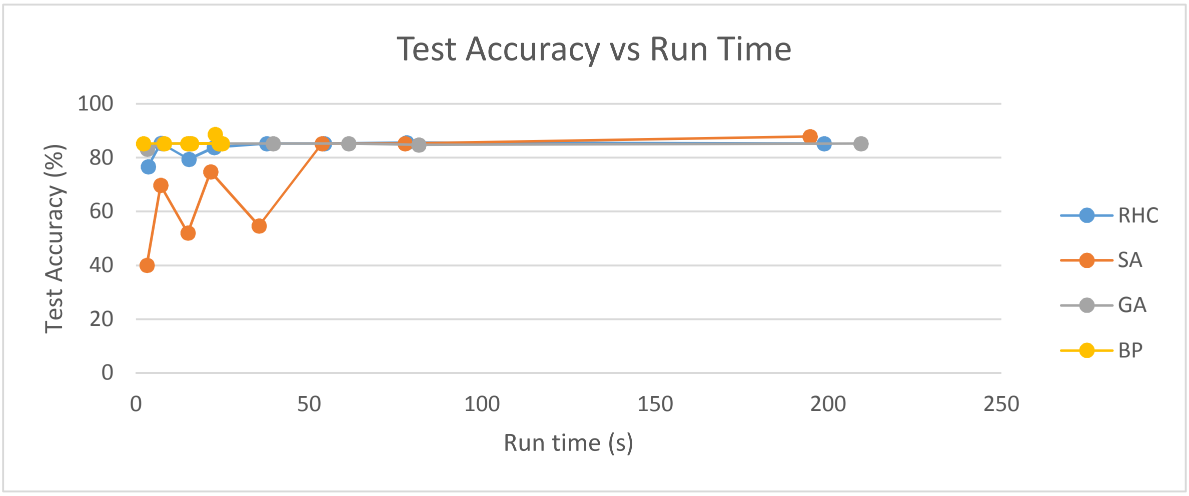

Figure 3. Test accuracy vs Run time for four different algorithms.

Table II
GA performance with different parameters

(population, crossover, mutation)	100 iterations	200 iterations	400 iterations	600 iterations	1000 iterations
(50,5,20)	85.202	85.152	84.091	85.051	85.202
(50,20,5)	83.99	85.051	85.202	85.253	85.455
(50,5,5)	82.626	83.384	84.949	85.152	83.838
(10,5,5)	69.293	79.343	57.677	69.646	84.545

COMPUTATIONAL COMPLEXITY

We are also interested in how the optimization algorithms handle problems with different data size and model complexity. We vary the model with input training data size and the number of hidden layers. We record the required number of iterations and runtime to reach the local maxima that yields 85.202 % accuracy. As you can see from Figure 4, with a small number of hidden layers, all four algorithms require small amount of iterations and run time to converge. However, as the model complexity grows, the required iterations and run time of RHC and SA also tend to increase. While RHC grows linearly, SA grows exponentially. I believe this is because as the model complexity grows, the number of local maxima also increases. Meanwhile, we set the start temperature very high so in the beginning the algorithm will behave like random walk. Therefore, it is very hard for SA to reach the local maxima above the threshold. The required run time and iterations of RHC also increases because RHC trust data completely and it will accept any improvements. As the number of local maxima increases, RHC has less chance to reach the local maxima that is above the threshold. Given more time however, RHC with less randomness compared with SA can reach the threshold faster than SA. GA and BP require less iterations and run time when give more complex models. Both algorithms soon reaches the maximal above threshold. We set the mutation size smaller than crossover size, so that it relies less upon local search and take more advantage of good restart points. Therefore, GA has a higher chance to detect the optimal local maxima than RHC and SA which simply relies more upon randomness and local search. As stated in the previous section, the dataset we use is well formed. Therefore, simply computes the gradient of a loss function with respect to all the weights in the network can directly reach the optimal solution.

There is no clear relationship between training set size and run time. As you can see the different levels of run time and iterations, the result is consistent with the rank set up in the previous section. There might be a relationship between training set size and run time for SA but there are not enough data points to argue for this statement. I would guess that more training data gives out more local maxima so that SA requires more time to converge to the optimal local maxima.

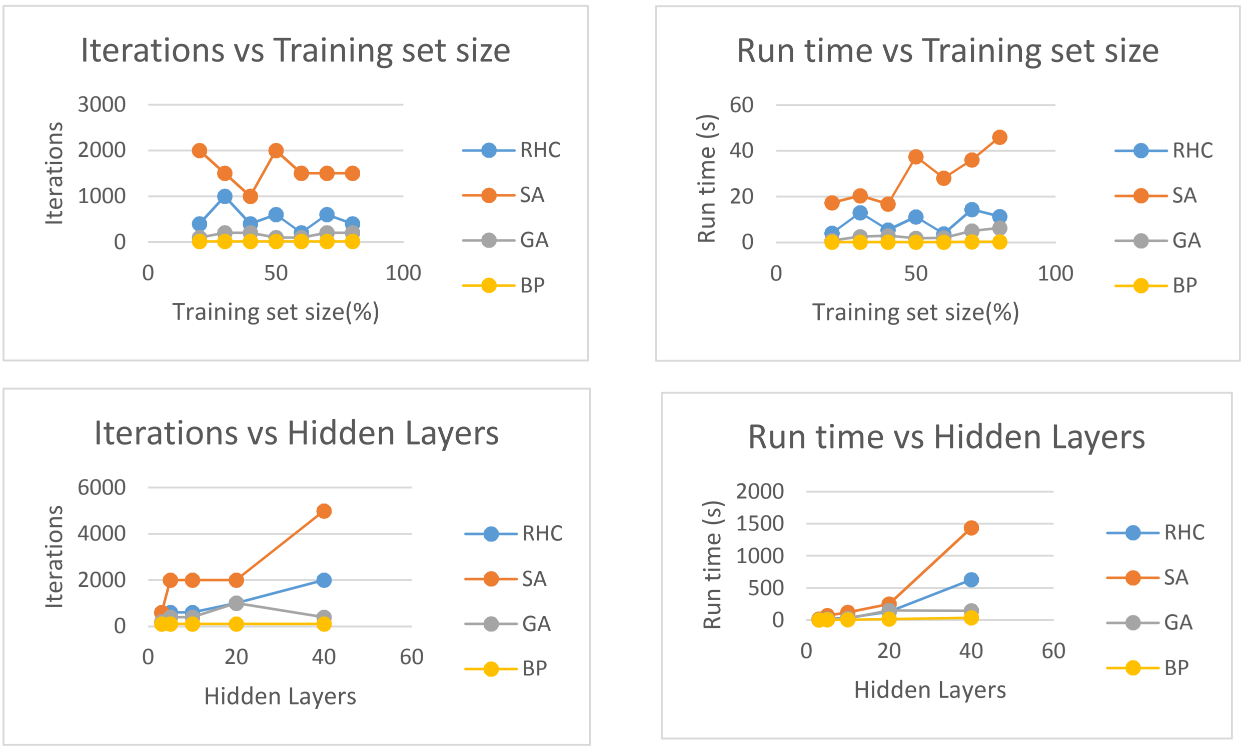

Figure 4. Computational complexity analysis.

OPTIMIZATION PROBLEM

APPROACH

To compare four algorithms, RHC, SA, GA, MIMIC, we run these optimization algorithms on three general problems: Four Peaks (FP), Traveling Sales Man (TS) and Max K coloring (MK). First, we evaluate the fitness of evaluation function versus the iterations and run time. This can help us understanding the performance of each algorithm with respect of time. Secondly, we change the systematic parameters like input sizes, cooling rate of SA, crossover size of GA, and retain size of MIMIC. We evaluate the fitness and the time it requires to converge to the maximal. In this way, we stretch out capabilities of each algorithm and to see how well it can perform at different conditions. Finally, we then compare their performance with respect of different metrics mentioned above to rank and draw the conclusion.

FOUR PEAK PROBLEM

There are two global maxima for the FP function. There are also two suboptimal local maxima. It has two parameters N and T and T is set as 20% of N. For large values of T, this problem becomes increasingly more difficult because the inferior local maxima has larger the basin of attraction.

When analyzing fitness versus run time we set a large input size in order to how each algorithm performs given a sufficiently complex problem. The results are shown in Figure 5. The iterations of GA and MIMIC is scaled down up 10 and 250 respectively in order to draw the fitness vs iteration graph on the same scale. You can read the unscaled version from Table III. Both RHC and SA converge instantly. This is because there are only a small amount of local maxima. Both RHC and SA have a large chance of starting at a good point so that it moves toward the global maxima. It takes many iterations of GA to converge to the global maxima because we set a high crossover value. Using crossover, we assume the population holds the information that can guide us to the global maxima. However, the distribution of population in this problem does not guide us to the right direction. After all there are only small amount of local maxima. We end up with random restart and local search based on a small mutation size and this can takes many iterations to reach the global maxima. The MIMIC, unlike the Professor's paper, actually yields the worst result which takes more than 100s to converge. I believe it is probably because the Four Peak problem relies much more on locality of sample instead of probability of samples. Samples are less correlated and therefore it is hard to model the probability distribution.

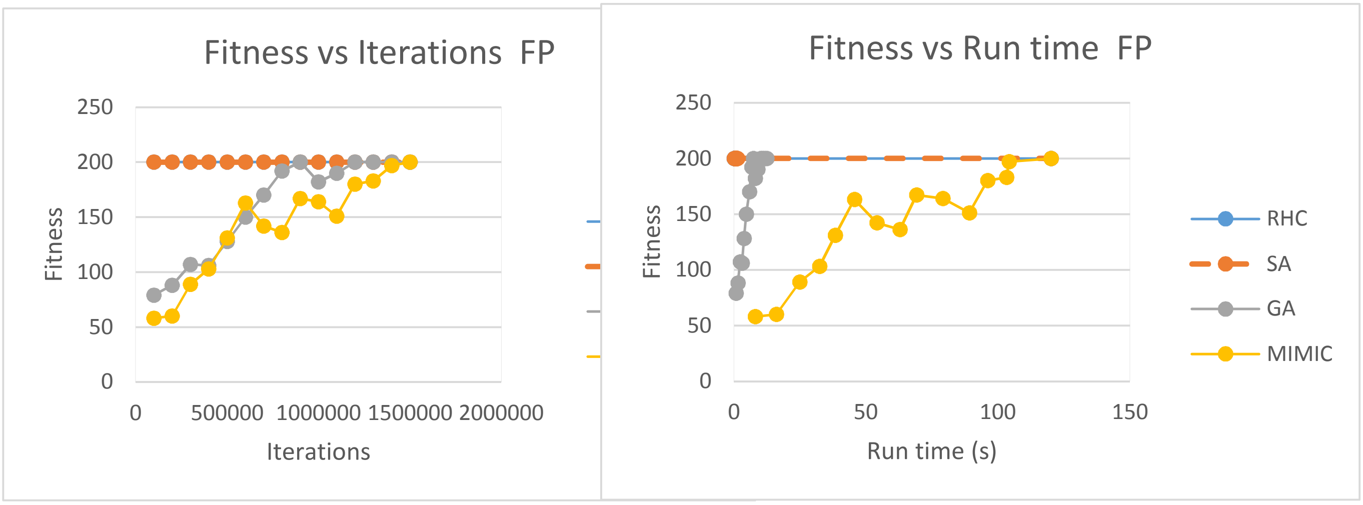

Figure 5. Fitness vs iterations and run time (Four Peak).

We are also interested in how each algorithm performs as the computational complexity grows. The results are shown in Figure 6. We test input sizes up to 200. For large values of T, the lower local maxima are larger due to the basin of attraction. However, different from Professor's paper, when evaluated with run time instead of number of iterations, MIMIC requires more time to converge to the global maxima. The last data point of MIMIC in Figure 6 is just the last data point of the graph of Figure 5 on the right. As we have large data size, MIMIC tends to spend more time than the other three algorithms. Although the required number of iterations of MIMIC is small as shown in Table III, it actually has very expensive steps and takes more time.

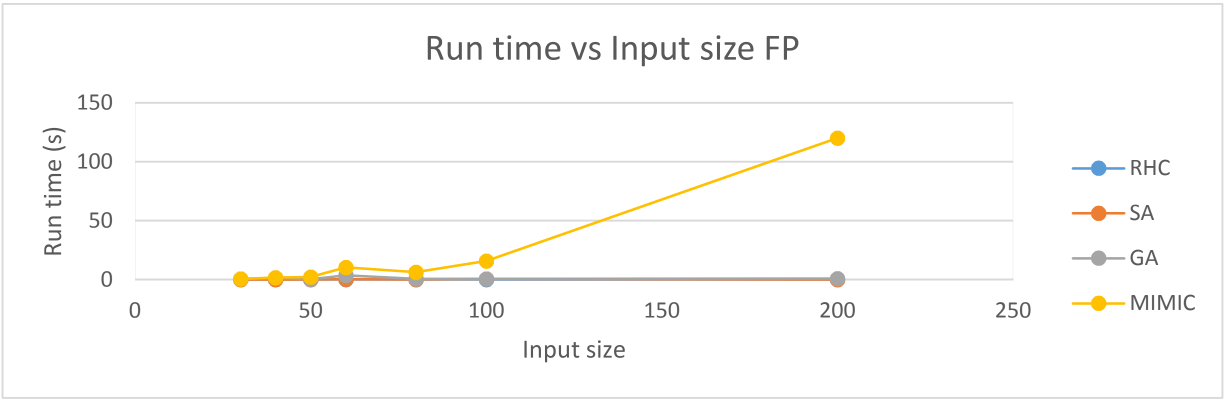

Figure 6. Run time vs input size (Four Peak).

Table III
GA and MIMIC iteration scale down with fitness

Input size	30	40	50	60	80	100	200
GA iterations	10000	30000	10000	70000	10000	10000	10000
GA fit	53	71	50	60	80	100	200
MIMIC iterations	400	1200	1200	4400	1600	2400	6000
MIMIC fit	53	71	89	107	80	100	200

TRAVELING SALES MAN

Given a list of vertices and edges, what is the shortest possible route that visits each vertices exactly once and returns to the origin city? In this problem, we use similar approach from FP section. The result of fitness versus runtime is shown in Table IV since it is hard to draw it on the same scale. As you can see from the Table, the run time of RHC and SA are comparable. RHC is an inexpensive algorithm in nature and SA in this case also has small run time because it behaves like RHC when temperature goes down. SA also yields highest fitness because its additional randomness helps us to jump over some local maxima that RHC does not. GA and MIMIC both spends long time even with scaled number of iterations. Their performance is slightly worse than RHC. I believe this is because the given vertices does not have probability relationship with each other so either crossover or dependency tree does not make sense in this case. TS may actually has local maxima very close to each other and therefore SA help use jumping over these basins to improve the performance.

Table IV
Fitness vs Iterations and Run time TS

RHC	Iterations	100000	200000	300000	400000	500000	600000	700000	800000	900000	1000000
	Training Time	0.104	0.209	0.321	0.444	0.508	0.617	0.679	0.757	0.865	0.969
	Fit	0.0761	0.0772	0.0833	0.0848	0.0868	0.0941	0.0855	0.0915	0.0826	0.0781
SA	Iterations	100000	200000	300000	400000	500000	600000	700000	800000	900000	1000000
	Training Time	0.21	0.406	0.613	0.835	1.035	1.248	1.352	1.529	1.7	1.899
	Fit	0.0913	0.0745	0.0895	0.09	0.0909	0.0941	0.0917	0.0964	0.0838	0.0897
GA	Iterations	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
	Training Time	1.763	3.567	5.382	7.141	9.008	10.533	12.377	13.149	15.43	17.584
	Fit	0.0627	0.0684	0.0692	0.071	0.0571	0.0696	0.0704	0.072	0.067	0.0717
MIMIC	Iterations	400	800	1200	1600	2000	2400	2800	3200	3600	4000
	Training Time	36.174	71.873	108.97	144.698	180.855	194.168	227.839	261.501	291.832	325.237
	Fit	0.05	0.0577	0.053	0.058	0.0587	0.0706	0.06227	0.071	0.0605	0.0733

MAX K COLORING

A graph is K-Colorable if it is possible to assign one of k colors to each of the nodes of the graph such that no adjacent nodes have the same color. The adjacent nodes per vertex and possible colors are fixed in my implementation. To analyze the fitness versus runtime, we also fix the number of vertices. The result is shown in Figure 7. SA and RHC seldom find suitable k coloring graph and therefore they have small fitness value. GA and MIMIC always find the k coloring and therefore they have the same high fitness value. The run time of GA however, is smaller than MIMIC. This is because MIMIC as relative expensive step while GA runs faster when set proper population sizes.

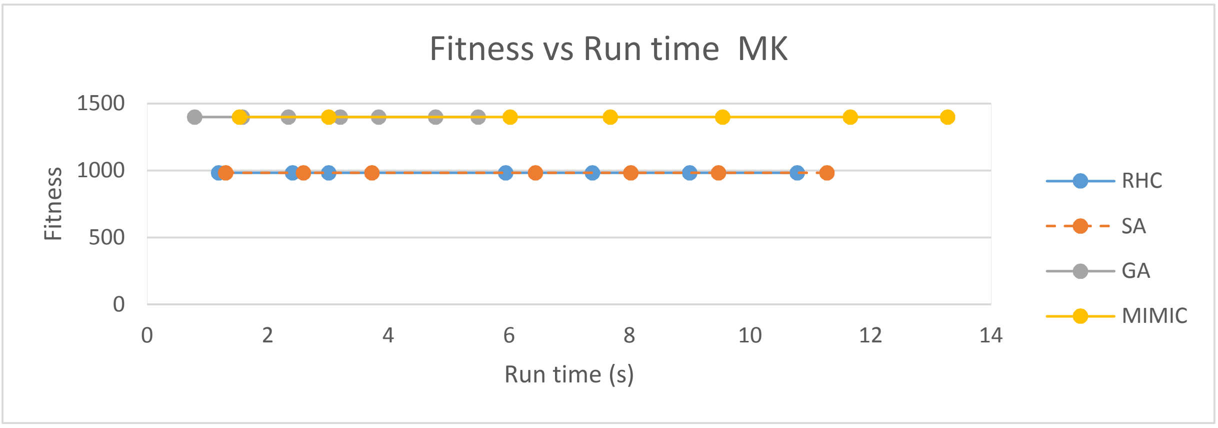

Figure 7. Fitness vs Run time (MK).

We also analyze how each algorithm deal with different input sizes. Input size here we define as the number of vertices. The result is shown in Figure 8. The run time of RHC and SA are very similar. This means that SA behaves almost like RHC. The cooling rate we set here is 0.1. In other words, it cools down very quick and therefore when temperature close to zero, SA is just like RHC. GA in this problem do not handle input sizes. It may result from the complexity of this problem. As we have more number of vertices, the population size does not change. Therefore, we need much more iterations to cover all possible routes.

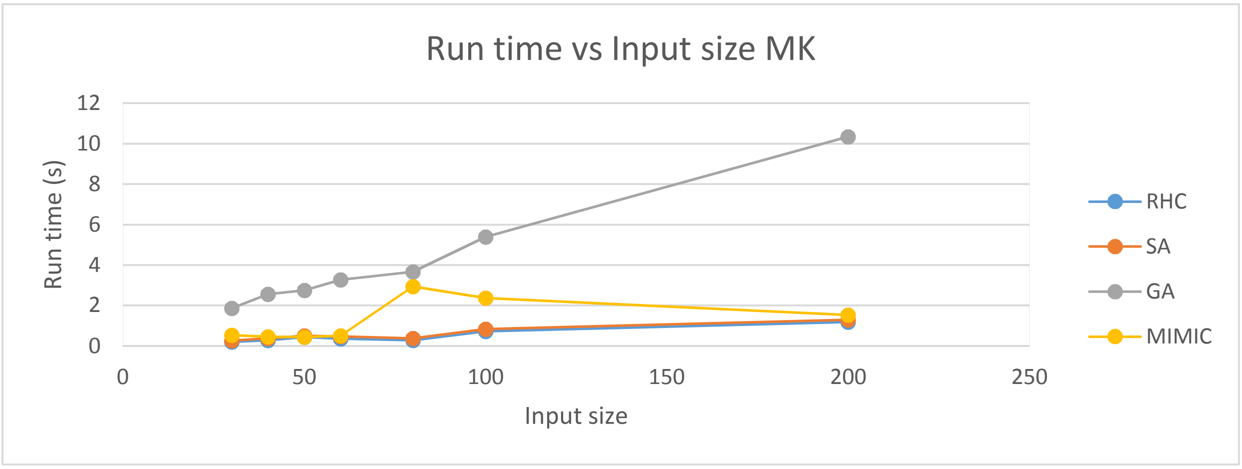

Figure 8. Run time vs Input size (MK).

PARAMETER TUNING

As stated in the beginning of section, we also vary the parameter of each algorithm in order to show how it behaves to each problem. It turns out that their behaviors are quite consistent over three different problems. The result is shown in Table V. The table shows that the cooling rate of SA does not have a strong impact on the fitness. As long as finding the precise global maxima is less important than finding an acceptable local maxima in a fixed amount of time, SA may be preferable. Crossover size of GA also does not have strong impact on fitness given enough amount of iterations. Smaller crossover size can decreases the run time for all three algorithms. This is because, high crossover size takes more time to finish and it also pair up more "good fit" individuals and more individuals will left after each iteration. The retain size of MIMIC does have a visible impact on fitness. The retain size is the maximum number of samples it retains after each iteration. There exists some optimal value for retain size to produce a higher fitness value. This value is also problem dependent, different problems may prefer different retain size. Smaller retain size also takes less time to finish. Obviously, less sample retained each iteration will speed up the progress.

Table V

Parameter tuning for three problems and four algorithms

SA	Cooling rate	0.95	0.7	0.5	0.3	0.1
FP	fit	80	80	80	80	80
	time	1.086	0.976	1.447	1.464	1.455
MK	fit	260	260	260	260	260
	time	4.344	4.4044	5.334	5.289	5.715
TS	fit	0.0925	0.0823	0.0947	0.09	0.0914
	time	2.752	2.698	2.629	2.65	2.754
GA	crossover	160	120	100	80	40
FP	fit	80	80	80	143	80
	time	12.79	10.28	9.141	8.147	5.474
MK	fit	518	518	518	518	518
	time	170.218	144.332	124.409	101.188	74.715
TS	fit	0.107	0.117	0.123	0.115	0.024
	time	128.598	95.704	83.549	63.333	31.264
MIMIC	Retain size	150	125	100	75	50
FP	fit	11	13	31	35	37
	time	76.058	63.571	52.648	46.207	34.275
MK	fit	427	463	506	506	463
	time	8.022	8.313	8.604	8.413	7.724
TS	fit	0.056	0.0687	0.0646	0.0791	0.0659
	time	509.507	484.162	468.269	454.309	443.217

CONCLUSION

Generally, we first analyze three optimization algorithms and BP over Neural Network on musk dataset. The summary and ranking is shown in Table I. We will summarize and rank four optimization algorithms over three general problems as shown in Table VI.

In the FP problem, based on my analysis, both RHC and SA performs almost the same and they produce and converges to the highest fitness within very small amount of time. GA and is better than MIMIC because it also takes small amount of time to converge to the optimal maxima, nearly as RHC and SA and it deals with increasing input sizes very well. MIMIC however, given a large input size, takes a long time to reach the optimal maxima although it finally produce the same fitness value.

In the TS problem, SA produces slightly higher fitness value than RHC because it jumps over local maxima which are close to each other. Meanwhile, GA and MIMIC preform bad in this problem in terms of run time. Given a large number of vertices, MIMIC takes hundreds of seconds to finish and GA is only slightly better than that.

In the MK problem, because the fitness value is mostly determined by whether the algorithm can find the K-coloring, RHC and SA is unable to find a proper K-coloring almost at every iteration. Although they attempt to optimize the situation, their results fail to satisfy the golden rule of this problem. GA and MIMIC both can find K-coloring and therefore they have higher fitness value. Moreover, GA runs even faster than MIMIC, so it is slightly better than MIMIC in terms of run time. However, GA fails to handle increasing number of vertices and MIMIC still converges within a small amount of time. Therefore, MIMIC is better than GA in this problem.

Table VI
Rank for optimization algorithms on three general problems

Algorithms	Rank for FP	Rank for TS	Rank for MK
RHC	1	2	3
SA	1	1	3
GA	3	3	2
MIMIC	4	4	1