Exploring Randomized Optimization Algorithms

Introduction

Randomized optimization algorithms are useful for many not well structured global optimization problems whose function often are not continuous or differentiable. Typically random search algorithms do not guarantee to find global optimal, but can find a fairly good solution quickly and some with convergence to find global optimal. In this analysis, we are particular interested in four of such randomized optimization algorithms, namely, randomized hill climbing(RHC), simulated annealing(SA), genetic algorithm(GA) and MIMIC. One feature these methods share is the use of probability in determining their iterative procedures, but they differ greatly in the way they explore the solution space. RHC and SA put great emphasis on the local search without take the structure or relationship of the underlying input parameters. In contrast, GA try to explore the structure through encoding the parameters into chromosomes and preserve and propagate the good groups through crossover. The result largely depends on the choice of encoding schema. MIMIC at the same time is also population based, but does not bear the same burden since it explore the space without predefining structure and it efficiently use what it learns from before.

This analysis focuses on exploring strength/weakness and some characteristic of these four algorithms. First we will try RHC, SA and GA on a classification problem that we discussed in our first assignment, specifically, we tested neural network with back propagation in first assignment--the IPO performance, and here we will use these three algorithms instead of back propagation to find the weight for the neural network. We preserve the same structure of the neural network with 3 nodes in just one hidden layer. The second part of this analysis will explore how SA, GA, MIMIC performs for three well defined optimization problem, and how it is related to some of its characteristics. The three problems are Traveling Salesman Problem (TSP), the Knapsack Problem, and the Four Peaks Problem. We used ABAGAIL implementation of RHC, SA, GA, and MIMIC and adopted the implementations of the three test problems in ABAGAIL as well.

Next, we will give a brief overview of the four randomized optimization algorithms and then proceed to our analysis.

Overview of Four Randomized Optimization Algorithms

1. Randomized Hill Climbing (RHC)

Randomized Hill Climbing searches for the minimum by randomly picking a point and moving toward the next higher neighbor(assume to maximize), until it reaches a maximum. Due to randomness (whether it is selection of starting point and which direction to go), global maximum cannot be guaranteed, and therefore the algorithm often finds local optimal very quickly and need technique such as restart to get better performance.

2. Simulated Annealing (SA)

Simulated Annealing algorithm uses probability to decide in what direction it will search towards in order to find the global maximum, modeling the slow cooling process of metals. It often chooses paths that are initially worse than the current point to exploring larger space to avoid weakness in hill climbing, therefore increase the probability of finding the global maximum. Theoretically, for any given finite problem, the probability to find the global optimal approached to 1, but it is not practically helpful since the time required to ensure such success will be close or exceed that of complete search of the space.

3. Genetic Algorithms (GA)

Genetic Algorithms is based on process of nature selection and tries to iterate over the population for a certain number of generations. The individual in the population is evaluated for its fitness to survive via fitness function. It explore the structure of the problem space to certain degree as determined by how it encode the problem in chromosomes, how the crossover/mutation be applied to create next population. It finds good candidate fair quickly, but it has tendency to converge to local optimal, which largely depends on the representation (chromosome encoding) and shape of fitness landscape.

4. MIMIC

MIMIC also tries to discover the underlying common structure about the space, but it does it by communicating information of the cost function obtained before for later search. In contract to GA, MIMIC does not put too much limitation by how the problem is structured and it use the prior knowledge gained more efficiently. it is more suitable for use on problems exhibiting higher correlation between input parameters. As it use the information gained before efficiently, it often achieve similar performance but with much less resource used (i.e. evaluation).

Training of Neural Network for IPO Performance Problem

In assignment 1 we trained neural network with back propagation for IPO performance problem. The weight selection is the optimization problem to maximize the predicating power of the network. Here we are going to see how the other three optimization algorithms, RHC, SA and GA compared when trying to solve the same optimization problem. We keep the neural network structure as the same as in assignment 1(with 3 nodes in one hidden layer). The training iterations is set to 1000 since our dataset is fairly small with fewer attributes, For SA we used larger T (1E12) and cooling rate 0.95

to allow thoroughly exploration of the search space. For GA we used 200 as population size and 100 to mate and 10 to mutate. Those will be considered as standard setting for the following discussion unless we note otherwise.

In table 1 we showed the final accuracy rate of the algorithm together with our baseline of back propagation and also there training time. As we can see, all three are in the same level of performance as back propagation, but slightly worse. Intuitively, this can be attributed to the randomness factor in those algorithms, as back propagation essentially doing gradient descent search, should find the global optimal if the problem space is well structured, as it seemed to be the case for our testing problem. As for training time, since GA is population based and we set the population to 200, it is not surprise that it need more time since there are more evaluation needed.

Algorithm	Accuracy	Train time in Seconds(1000 Iterations)
Back Propagation	85.58%	0.3
RHC	82.69%	0.771
SA	79.81%	0.694
GA	81.73%	9.717

Table 1 Performance on neural network training

We plotted and the error rate over time in figure 1 to show the behavior of the algorithms overtime due to randomness and how they converge. As we can see, all three algorithm shows convergence to lower error rate after certain steps of iteration, clearly demonstrate the signature of random optimization algorithms. However, each also clearly shows its own characteristic.

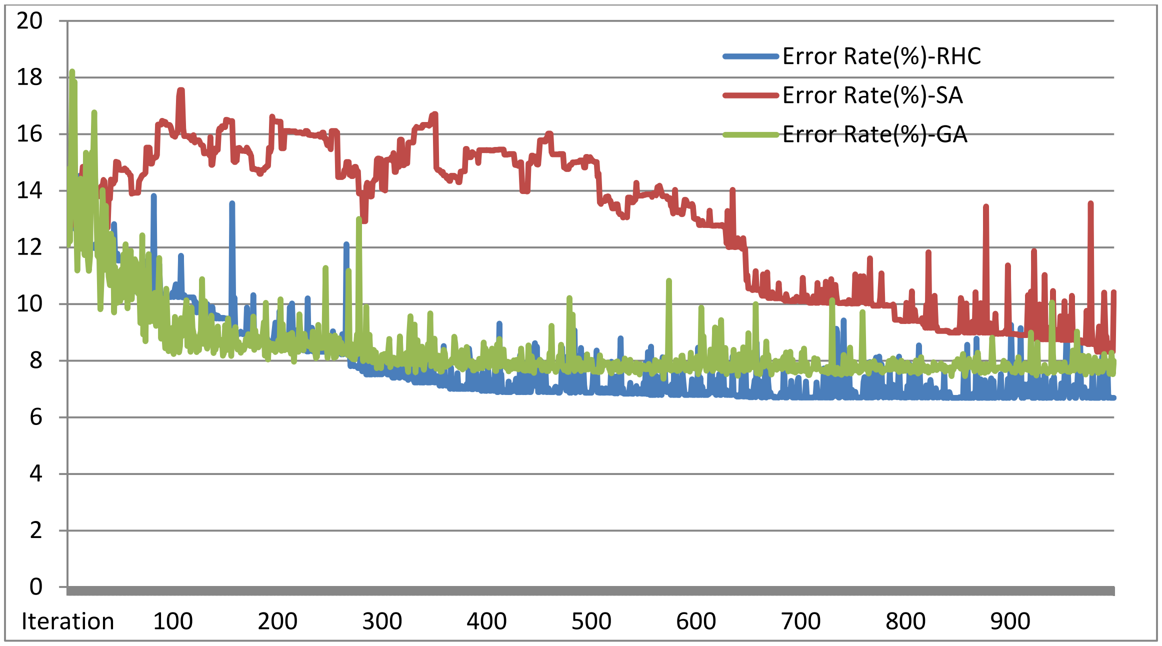

Figure 1 The error rate over 1000 iterations

From the RHC, we can see it converge fairly quickly and error rate is fairly stabilized after 500 iterations with few spikes which indicating a local random search deviation from the optimal. It also suggested our testing problem is most likely not ill-structured.

The GA converges much faster as it finds good candidates more efficiently, but as consequence of not fully exploring the search space, it settles in a less optimal solution. Encoding of chromosome could also be a factor here.

As for SA, we chose higher cooling rate and starting temperature, and the algorithm clear try to explore the space more and care less about performance at early stage as it is evident from Figure1. It coverages slowly before 600 iterations, and once it is stabilized at later stage, it can rarely deviate to get better result. This is clear demonstration of feature of SA.

To further explore how cooling rate impacts the search for SA, three different values for cooling rate were tested, as illustrated in Figure 2. As we can see, lower cooling rate leads to much faster convergence. As for this test problem, it also results better solution within given number of iterations. Higher cooling rate need more times to converge and spends great deal of time exploring the space before it limits its search to much smaller space. Given enough time, higher cooling rate could end up with better solution.

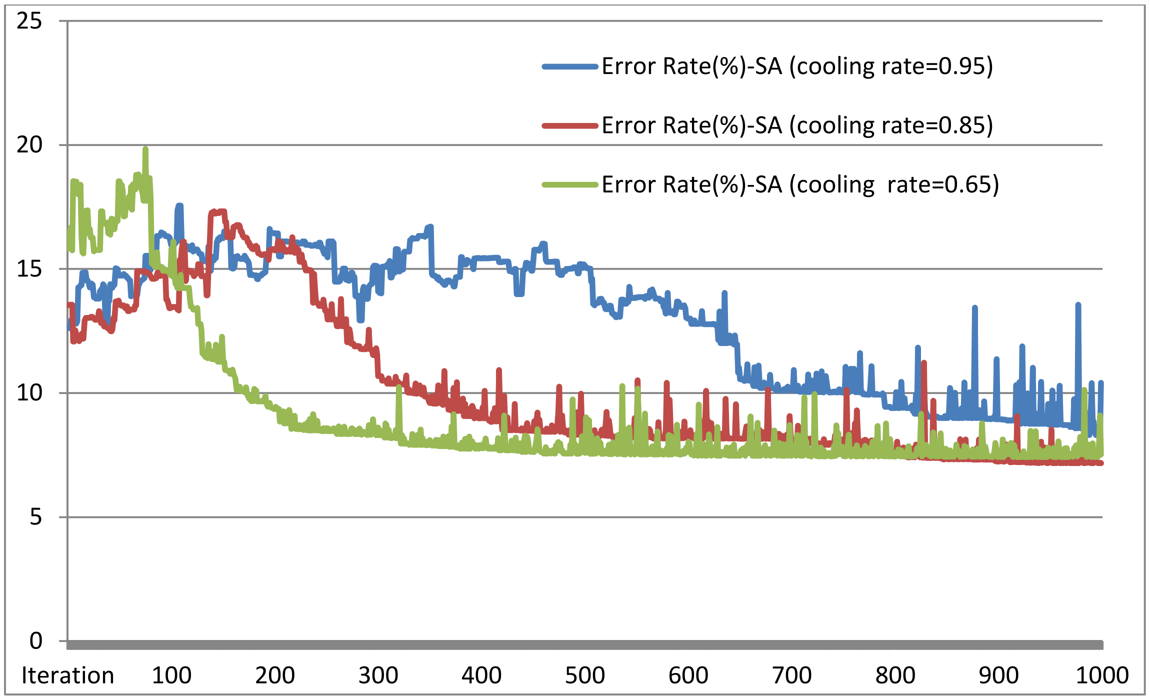

Figure 2 Error Rate for SA with different cooling rate

To see how different settings change the behavior of GA, we set the initial population to a smaller size to 100, and also explore when setting a much lowered mate ratio (from 100 to 20). The results are shown on Figure 3. Both has adverse effect; the small population exhibits similar behavior to our standard setting, but with less optimal results most likely the diversity of the population is not rich enough, in other words, it limits the search space thus the sub optimal performance. And we see even bigger issue if you use very lower mate ration, and there could two contributors for its bad performance, First, less mate meaning less chance to generate solution candidates, second, less mate also leads to poor propagation of good solution component to whole population. as it is evident from Figure 3, around 100 iteration, it seems has some good solutions, but it is soon dominated by some other inferior one, and until around 950 iteration, most likely due to mutation, it finds better ones, but we do not know if it will converge or not.

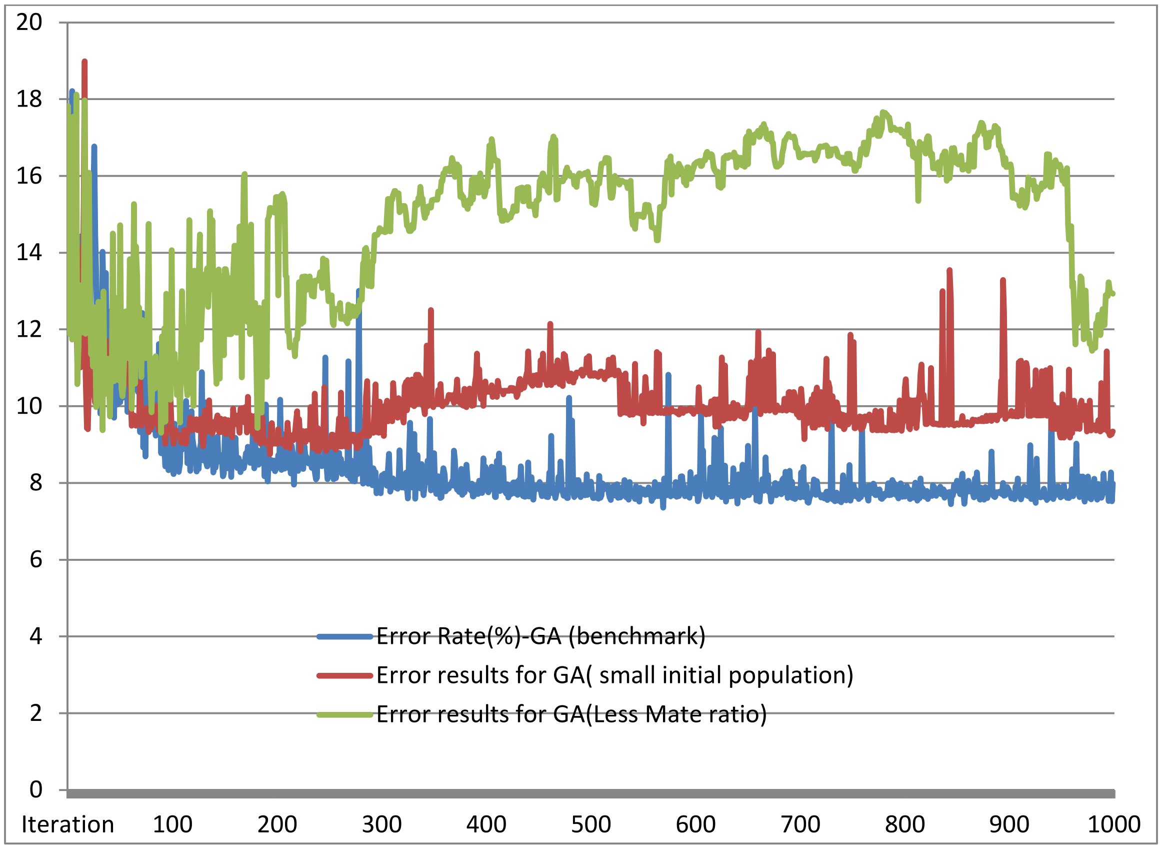

Figure 3. Error rate under different setting of GA over 1000 iterations

Testing on Three well known Optimization Problems

The three test problems we choose for explore the performance and characteristics of SA, GA and MIMIC are traveling salesman problem, knapsack problem, and four peaks problem.

1. Traveling Salesman Problem

The Traveling Salesman is a classic NP-hard problem wherein you need find the minimum path to visit every city and return to the starting city, given a number of cities and their connections. It is one of most intensively studied problem in optimization. The search spaces can be very complex and often has many local optimal. Intuitively one good solution can be a combination of good solutions for sub problems of same problem, for example, considering cluster of cities that is close to each other but is far from another cluster of cities, we probably can solve the TSP for this subset of cities and then combine to get a solution for the original problem. Since GA and MIMIC both has feature to explore the structure of the problem, we expect to see better performance from these two. The problem tested is randomly generate for with 50 cities and try to set it as maximization problem, we use 1/distance as the metric to measure the performance.

2. Knapsack Problem

The Knapsack Problem is a dilemma in which you have different objects with different weights and values and a sack in which you wish to carry the maximum value in your sack...The constraints and the evaluation function all are linear, but the problem is both NP-complete and NP-hard. But many cases in practices can beloved exactly. We generate our testing case randomly with a maximum weight and volume of 50 units per item, with 40 unique items of which you could have 4 copies of each. In most likely circumstances, we expect to see good performance among all the threes.

3. Four Peaks Problem

The Four Peaks problem is well designed problem to post challenge for algorithms like hill climbing, it has continuous local maximum values with fairly large plateaus that allow some algorithms to easily get stuck in the local optimal, thus fail converge to a better solution. As the parameter T becomes a larger percentage of N, the part of the space containing the global maxima becomes significantly smaller. So it becomes less and less likely that one could just fall into it by sheer luck. The search space is not continuous and differentiable and it will be a good candidate to see how the three algorithms can deal with the large plateaus. We expect it will be difficult for all three, but MIMIC and SA could have better chance coming up with better ones, For SA, we will rely on pure luck, there is small chance we could hit the jackpot, For MIMIC, it explore the shape of the whole space and it has superior knowledge and much better chance to end up well. For GA because the inherent genetic operator, the local optimal usually will dominate and unless we set everything perfectly, the better solutions will have much less chance to stand out. For the testing case we set T=N/10 and N=80.

We set the iteration for SA to 20000. To make it comparable, the GA population is set to 200 and the iteration is set to 1000, so is MIMIC. We run the test 10 times and collect the maximum found as benchmark. And we use the average of the 10 runs divided by the benchmark as the performance metrics for the algorithms. Figure 4 shows the relative performance for the three algorithms for the three testing problems.

For knapsack problem, it is in line with our expectation, as all three algorithms perform really well. MIMIC and GA come in close but better SA and it can be explained that the problem has certain parameter correlation that can be easily exploited by MIMIC and GA while MIMIC does it better.

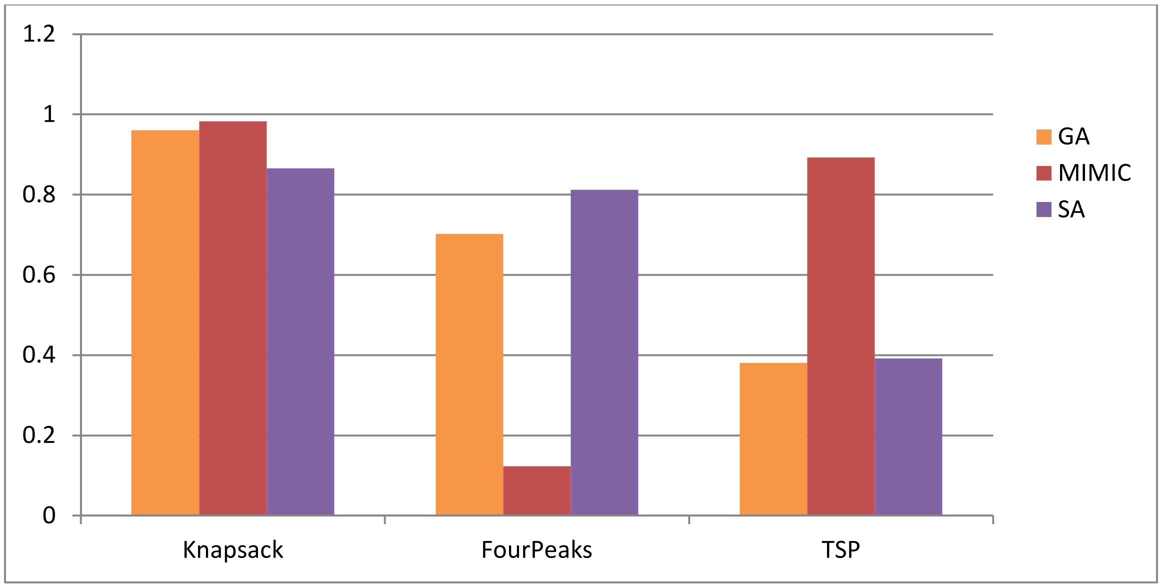

Figure 4 Performance of SA, GA, MIMIC on three problems

For TSP, MIMIC really stands out and demonstrates its strength in mapping the shape of the search space and its ability to learn from prior knowledge, while GA and SA suffer in performance but for different reasons. The poor performance of GA could be due to parameter settings or the encoding of chromosome that does not represent the problem well for GA. SA is most likely not well suited for such problem and is more prone to stuck with local optimal.

For Four Peaks problem, we are surprised to see SA performs much better than GA and MIMIC, and the result is especially shocking for MIMIC which we expect to do much better. So we decide to get MIMIC another chance by changing the parameter setting.

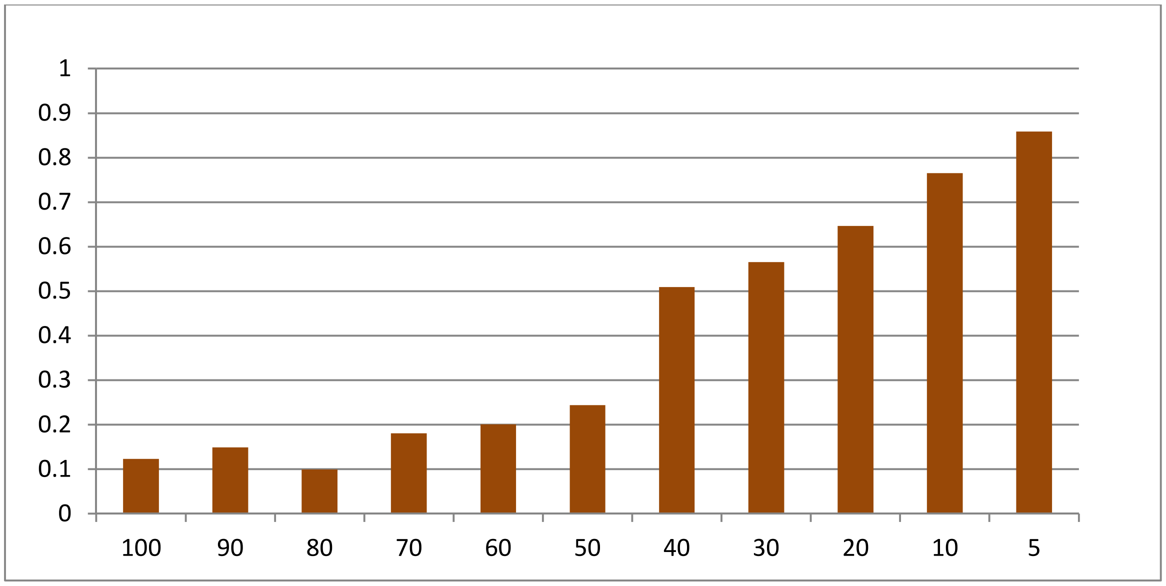

Figure 5 Performance of MIMIC with different # sample to keep for Four Peaks

First we change the number of sample keep in MIMIC from the standard setting of 100 to until 5, Figure 5 shows dramatic improvement when we decrease this parameter, and for value 5, MIMIC perform slightly better than SA(0.8589 for MIMIC vs 0.8119 for SA) . This suggests for MIMIC to work better on this kind of problem with large plateaus, the selection of sample should be stricter, otherwise, the mapping learned will be more plateau dominant and it will be much easier for MIMIC to be stuck in plateau.

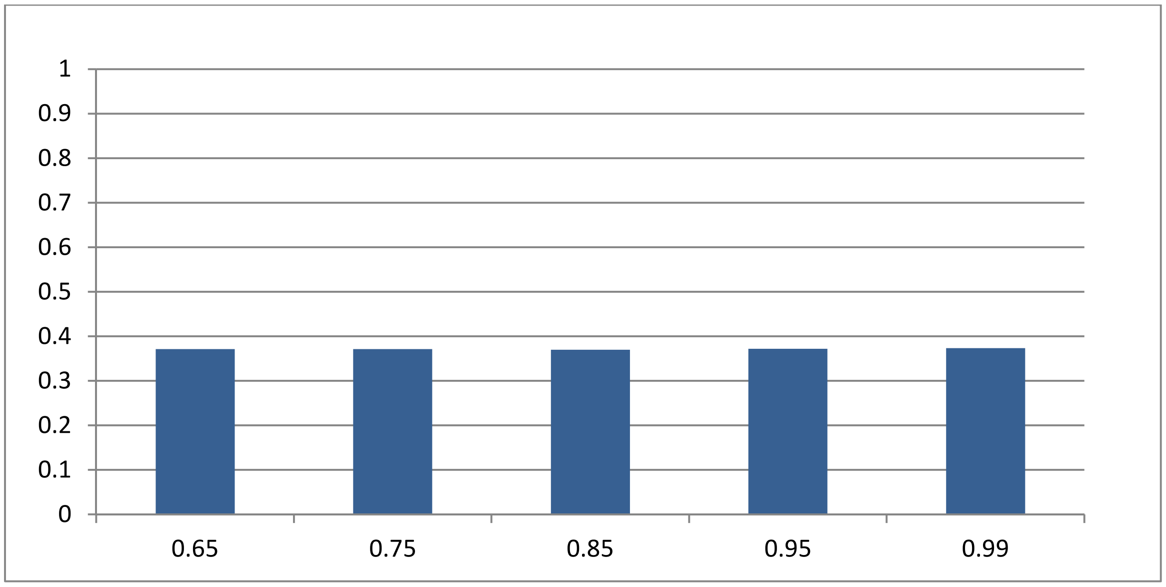

Figure 6 Performance of SA with different cooling rate for TSP

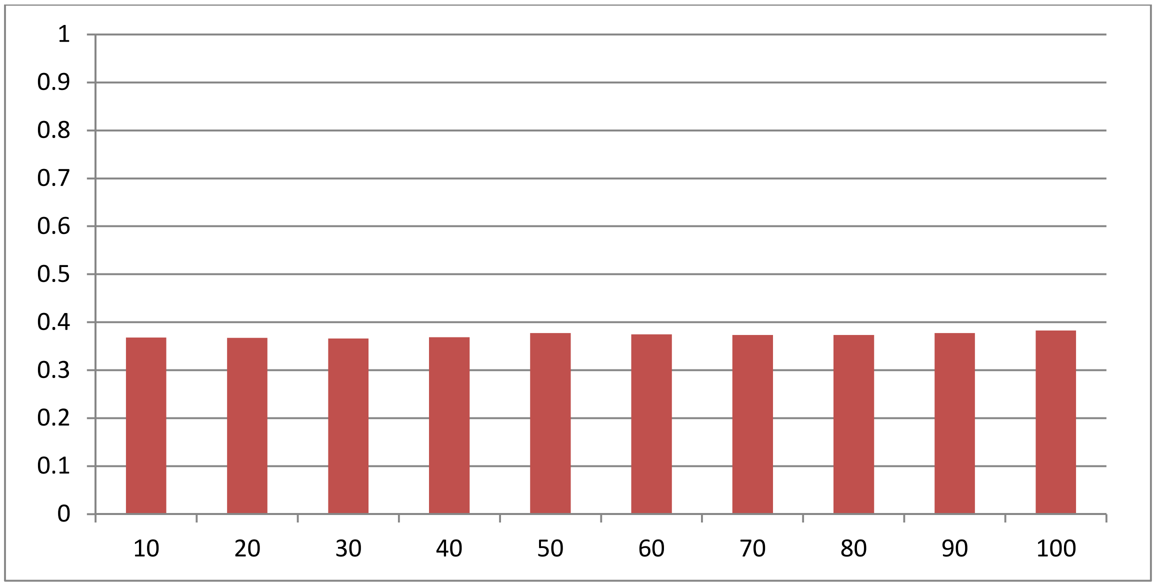

Figure 7 Performance of GA with different mate ratio for TSP

To give SA a fair chance to improve on TSP, we change the cooling rate from 0.65 to 0.99, Figure 6 shows no further improvement and it again confirms our thoughts that SA is easy to stuck in local optimal for TSP as this problem usually has many.

For GA, we would like to see if performance can be improved by changing the mate ratio. We set the number to mate from 10 to 100 and Figure 7 shows no improvement.   This further leads us to the conclusion that this is more due to not good enough representation of the TSP problem in GA's encoding choice.

Conclusion

As we can see from the two parts of empirical study, MIMIC in those testing problems performs better due to its learning of whole space and use of prior learned knowledge, it suffers less likely b/c it does not depend heavy on settings like GA 's encoding choice. Regarding to computational cost, it is most likely use less resource b/c of it is stochastic guided search. However, some problems are still going to be challenging to MIMIC and all randomized optimization algorithms because of characteristic of shape of search space (Four Peaks for MIMIC). There is no single best algorithm and even a better algorithm need to be adjusted to perform well on some problems, which often requires good understanding of problem itself.