1 Introduction

The purpose of the assignment is to compare and contrast the four optimization techniques. In the first part of the assignment I implemented three different problems using Random Hill Climbing, Simulated Annealing, Genetic Algorithm and MIMIC.

Optimization Algorithms:

The Optimization algorithms used are as follows. I would give a brief description and the strengths and the weakness of each algorithm

1. Random Hill Climbing : Randomized Algorithms: Randomized algorithms are algorithms that incorporate some form of randomness during computation to solve a problem. This randomness can help the algorithm explore different possibilities more efficiently and can also help in solving problems that are difficult or impossible to solve using deterministic algorithms. Randomized algorithms are used in various fields, including computer science, mathematics, and engineering.

2. Simulated Annealing : Simulated annealing is a randomized optimization algorithm that is inspired by the annealing process in metallurgy. In this process, a material is heated to a high temperature and then slowly cooled, allowing the atoms to arrange themselves in a lower-energy state. Similarly, in simulated annealing, we start with an initial solution and then iteratively perturb it by introducing random changes. The new solution is accepted with a probability that depends on the difference in the objective function value and a temperature parameter. The temperature is gradually decreased during the search process, allowing the algorithm to escape local optima and explore the solution space more effectively.

3. Genetic Algorithm : Genetic algorithms are a class of evolutionary algorithms that are inspired by the process of natural selection. In genetic algorithms, a population of candidate solutions is evolved over time using genetic operators such as mutation, crossover, and selection. The fitness of each solution is evaluated using a fitness function, and the solutions with higher fitness values are more likely to be selected for reproduction. Over time, the population evolves towards better solutions, providing a global optimization approach that can handle complex, non-convex objective functions.

4. MIMIC : Mutual-Information-Maximizing Input Clustering (MIMIC) algorithm finds optima by estimating probability densities. Most of the optimization algorithms lack structure and MIMIC uses knowledge of this structure as a guide for randomized search through the solution space. It attempts to communicate information about the cost function obtained from one iteration of the search to later iterations of the search directly.
   MIMIC implementation used in this assignment is again done using mlrose. I will attempt to take advantage of the structure MIMIC maintains in order to compare its performance against other optimization algorithms in the sections to follow.

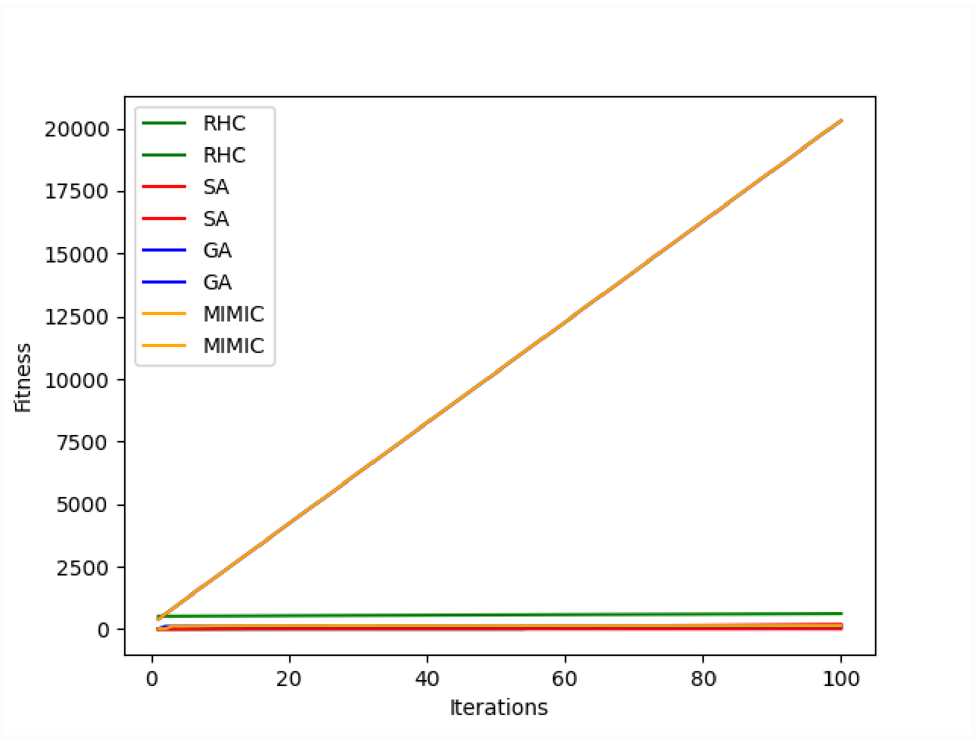
Optimization Problems:

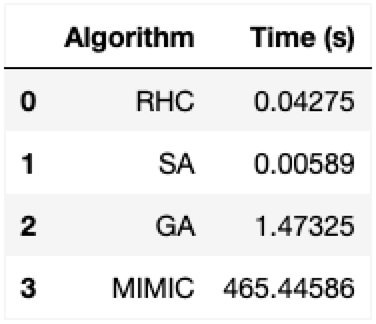
This section covers the optimization problems which are solved using the Random Optimization algorithms.

1. Continuous Peaks
2. KnapSack
3. Traveling SalesMan

Continuous Peaks : The Continuous Peaks problem is a scenario where the optimization algorithm needs to navigate a 1D space with many local optima, resembling the task of determining the elevation of multiple peaks. This problem is chosen for its simplicity and contains numerous local optima.

Below is the performance of the Four Randomized Algorithms using the mlrose_hiive library. MIMIC performed the best as shown in the picture below with highest Fitness values(Fig1). However, one of the downsides of using MIMIC is the execution time. MIMIC takes a long time to execute as depicted in the Execution times of each of the algorithm(Fig2)





	Algorithm	Time (s)
0	RHC	0.04275
1	SA	0.00589
2	GA	1.47325
3	MIMIC	465.44586

Fig 2

KnapSack Problem:

The Knapsack problem involves maximizing the value of items placed in a knapsack with a fixed capacity, W pounds, while considering the weight and value of each item. Each item i has a weight wi and a value bi, and the goal is to select quantities xi (0≤xi≤wi) of each item to maximize the total value within the weight constraint. There are two types of Knapsack problems: bounded and unbounded. In the bounded version, only a limited number of each item can be selected, while the unbounded version allows selecting multiple items of the same type. The problem can be solved using dynamic programming by calculating the optimal solution for different weights and items. The fractional Knapsack problem relaxes the constraint of selecting whole items, allowing for fractions of items to be included based on their value-to-weight ratio.

Observation:

Based on the fitness function and the best values for it the Genetic Algorithm performed the best as shown the Fig3. MIMIC algorithm was very close to the Genetic Algorithm. However the Genetic Algorithm performed better. The hyperparameters for the Algorithms were optimized. There is considerable performance cost for the MIMIC algorithm as depicted in Fig 4. However the RHC, GA, SA were tied in terms of the execution times for them. The Genetic algorithms mutation and population size were tuned for the best performance.

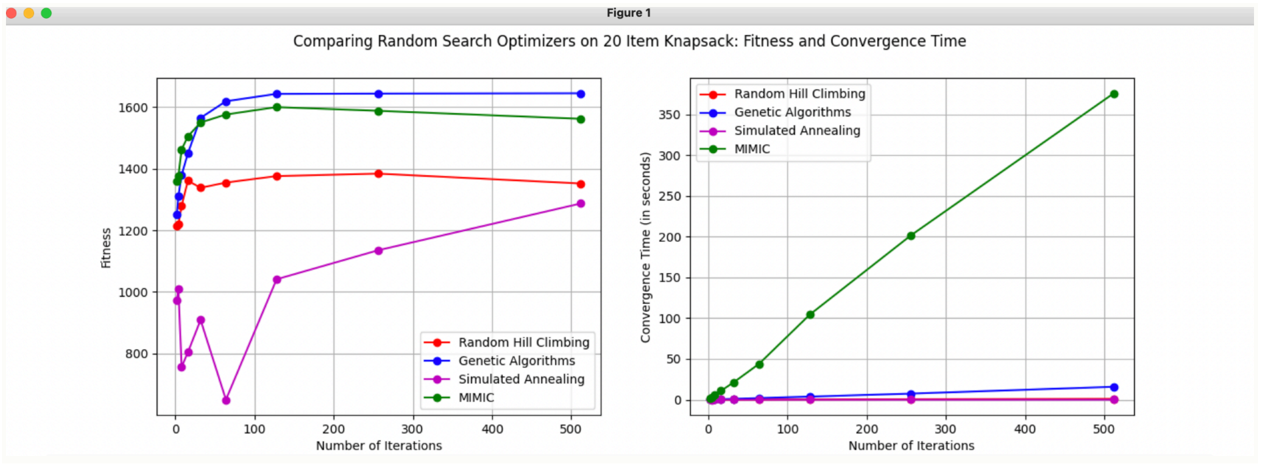


Fig 3 Fig 4

Traveling Salesman Problem:

The Traveling Salesman Problem (TSP) is a classic optimization problem where the objective is to find the shortest possible route that visits a set of cities exactly once and returns to the starting city. This problem involves determining the most efficient path for a salesman to travel between multiple cities, minimizing both travel costs and distance traveled. TSP is an NP-hard problem, making it challenging to evaluate every possible solution for larger instances. Various algorithms, such as dynamic programming, simulated annealing, and 2-opt, are used to solve the TSP efficiently. The problem has real-world applications in vehicle routing, circuit design, and network optimization, among others

Observations :

Based on the Fitness function the MIMIC algorithm was clear winner over the other algorithms followed by GA, RHA and SA.

Execution times : MIMIC takes a considerable amount of execution time.

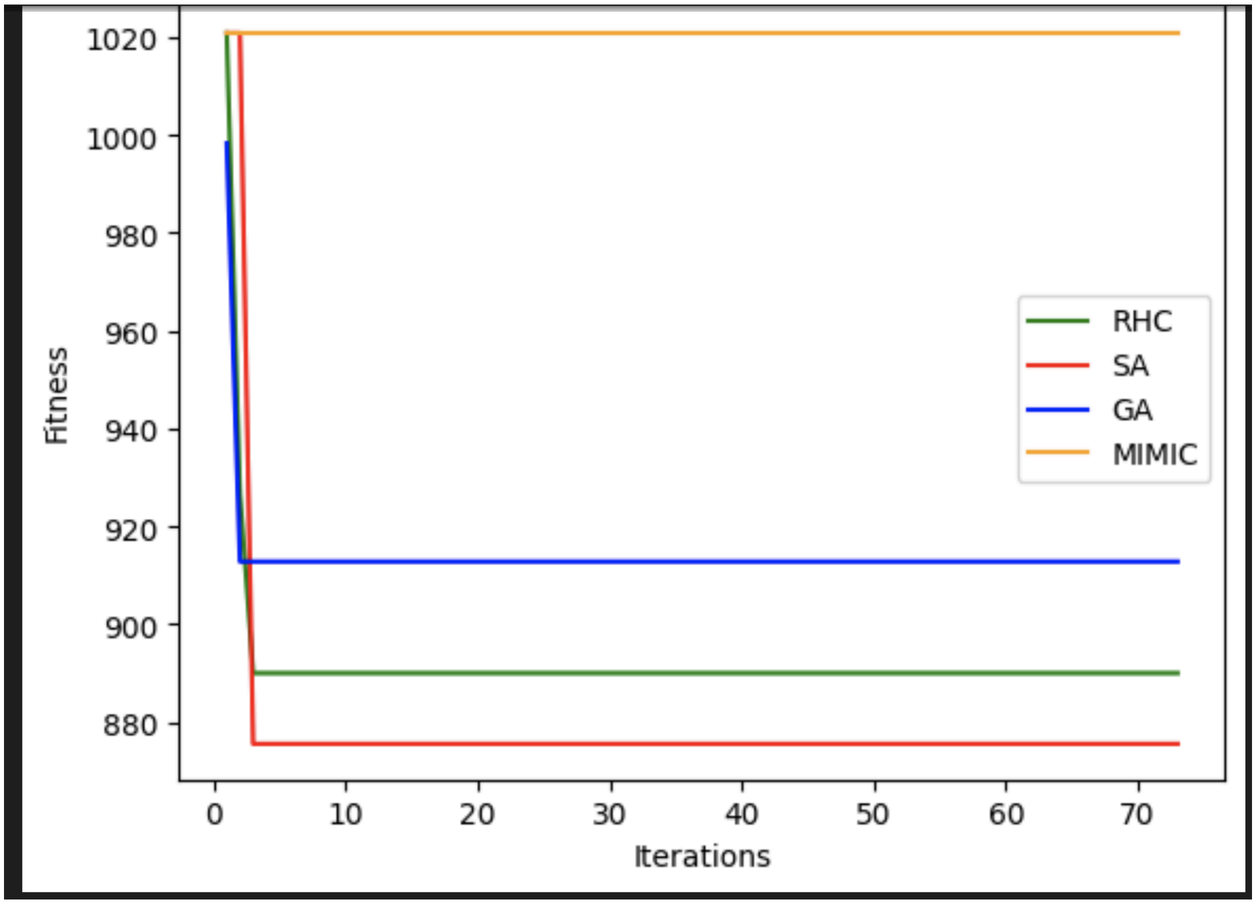
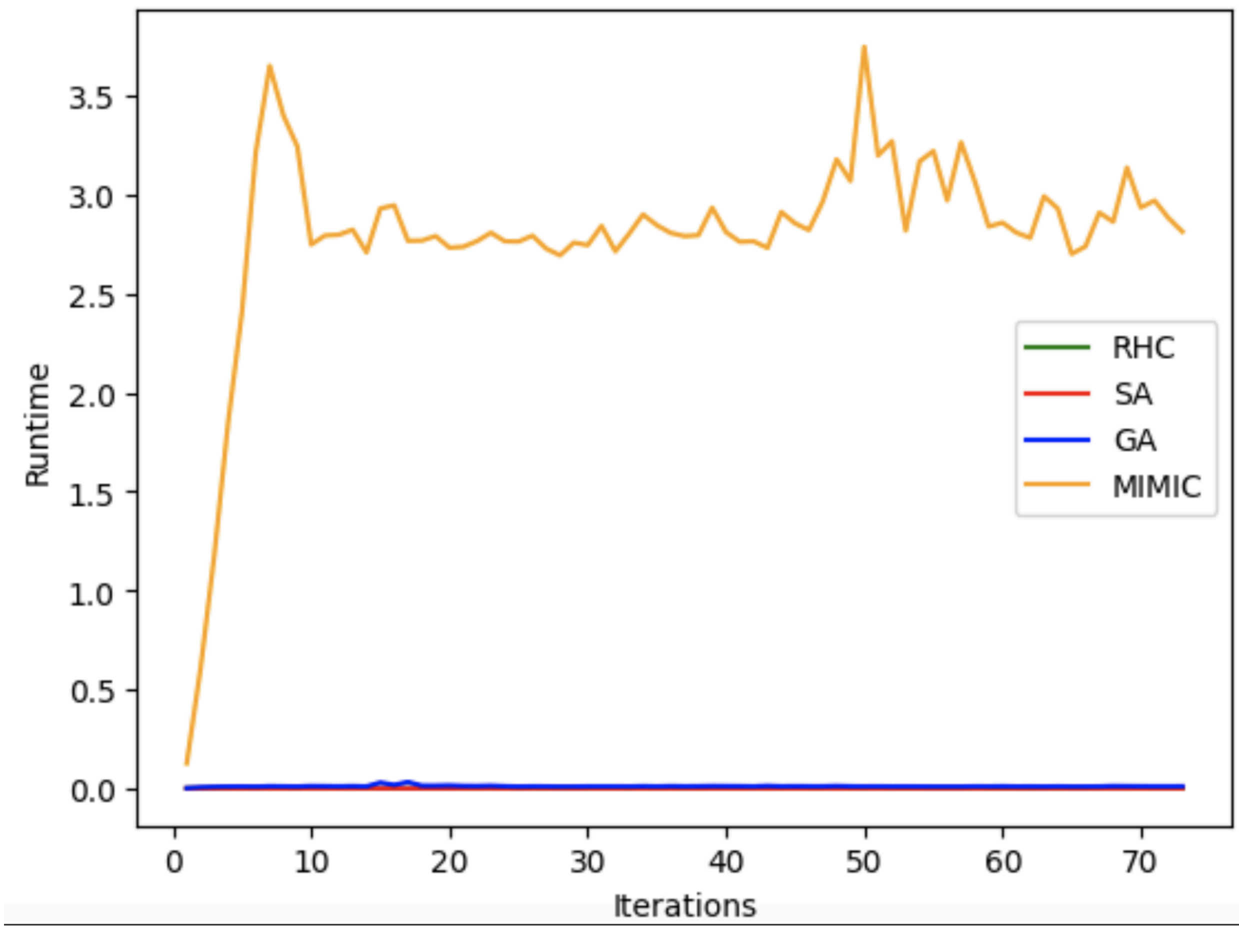

1020
1000
980
960
940
920
900
880
Fitness
0
10
20
30
40
50
60
70
Iterations
RHC
SA
GA
MIMIC

Fig 5



Runtime
Iterations
RHC
SA
GA
MIMIC
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50
60
70

Fig 6