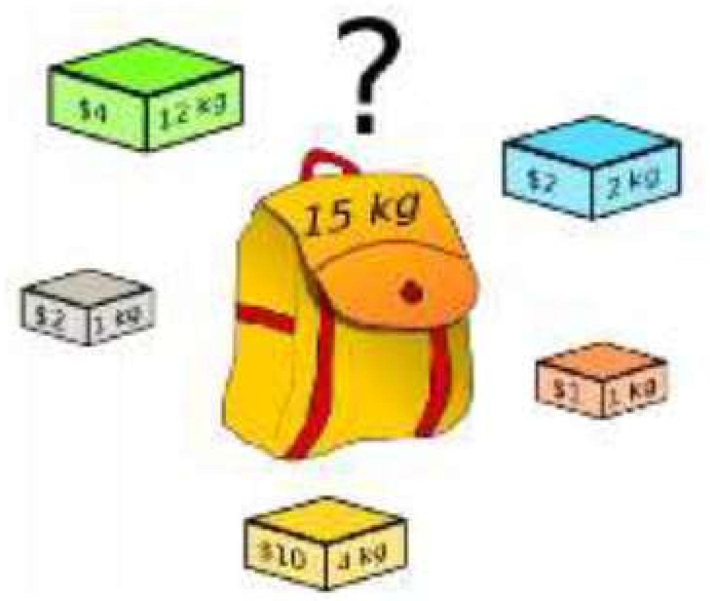
AI PROJECT REPORT II
KNAPSACK PROBLEM



Ege Kutay YÜRÜŞEN / 180316017

Oğuzhan Çevik / 1703016045

A-Problem Formulation

- State Representation: Our state is representation of an array(bag) that consists of 1's and 0's. Each array's cell represents an item. If the value is "1" then it means our bag contains the item. If it's "0"that means it doesn't contain that item.

- Possible Actions: Action selects each index in the array to be operated in result.

- Transition Model(result): The result for our problem reverses the value of given index. If the action is 2, it selects index 2 in the array (assuming array size is bigger than 2) the value in the index 2 will be reversed. Which means if the value is "1" it will convert to "0" or if the value is "0" to "1".

- Fitness/Objective Function (value): The items in the bag has two attributes which are weight and their worth. We store each of these attributes in the separate arrays. The value is calculated by sum of the items worth in the bag but if the items in the bag exceed weight capacity of the bag then the value of that state is 0.

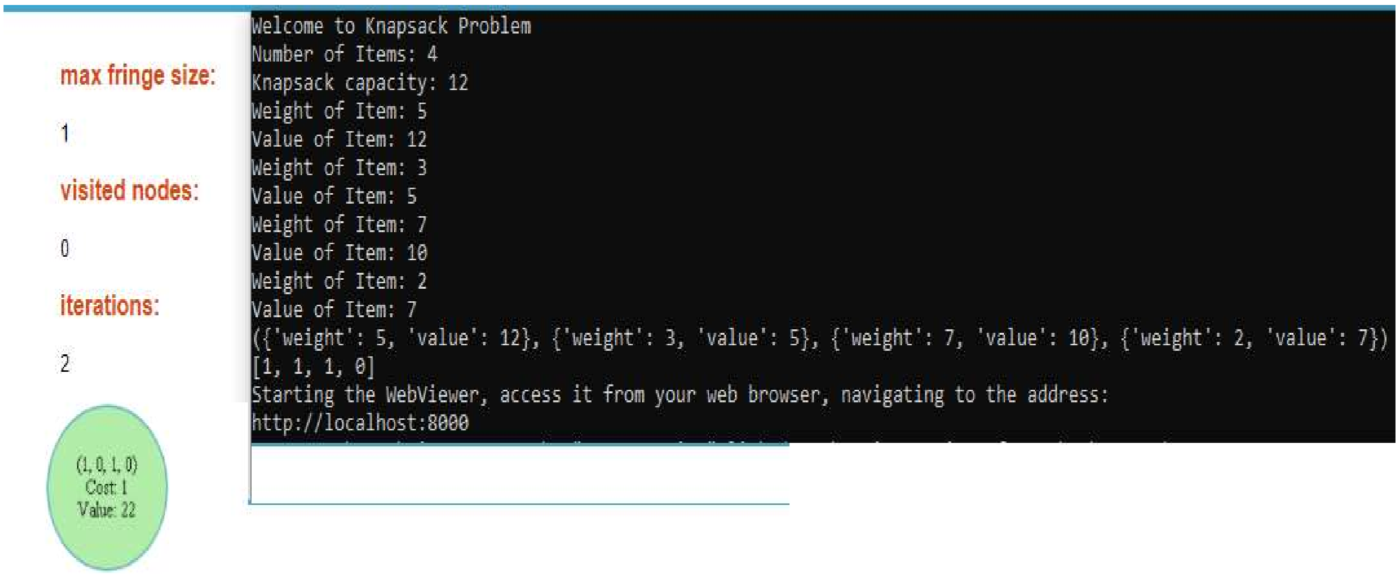
- Crossover: Cuts the randomly place of child's chromosome (in other words array index) and the chromosome being inherited from both parents in the random cut index.

- Mutation: One of the child parent chromosomes is generated randomly and the other parent chromosome remains same. Then parent chromosome combined in random index to generate child's chromosome.

B- Discussion on the results.

Hill Climbing Search:

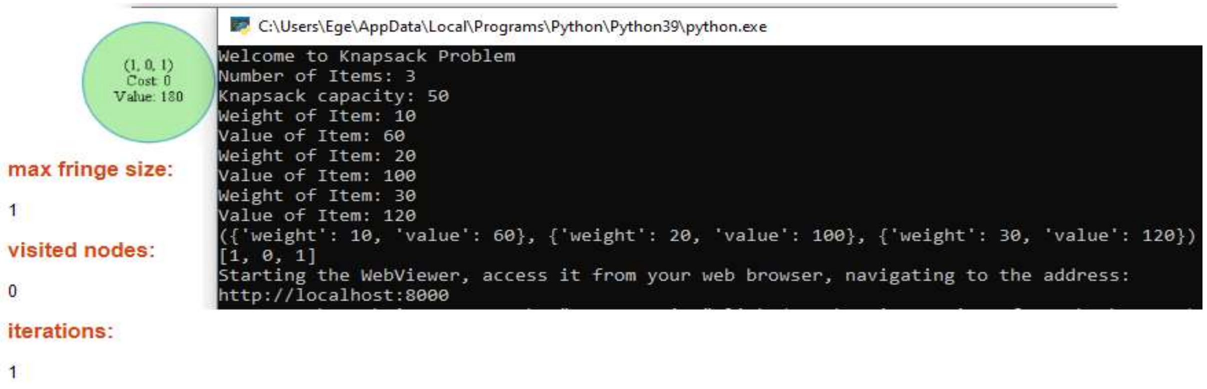
1)



```
Welcome to Knapsack Problem
Number of Items: 4
Knapsack capacity: 12
Weight of Item: 5
Value of Item: 12
Weight of Item: 3
Value of Item: 5
Weight of Item: 7
Value of Item: 10
Weight of Item: 2
Value of Item: 7
({'weight': 5, 'value': 12}, {'weight': 3, 'value': 5}, {'weight': 7, 'value': 10}, {'weight': 2, 'value': 7})
[1, 1, 1, 0]
Starting the WebViewer, access it from your web browser, navigating to the address:
http://localhost:8000
```

1) The example above, there is two available global maximums which are 0,1,1,1 and 1,0,1,0 both state's values are equal to 22 (optimal for given input). So, the hill climbing search found one of the optimal results.

2)



```
C:\Users\Ege\AppData\Local\Programs\Python\Python39\python.exe
Welcome to Knapsack Problem
Number of Items: 3
Knapsack capacity: 50
Weight of Item: 10
Value of Item: 60
Weight of Item: 20
Value of Item: 100
Weight of Item: 30
Value of Item: 120
({'weight': 10, 'value': 60}, {'weight': 20, 'value': 100}, {'weight': 30, 'value': 120})
[1, 0, 1]
Starting the WebViewer, access it from your web browser, navigating to the address:
http://localhost:8000
```

2) The example above the hill climbing search did not found the optimal solution and returned local maximum value, the global maximum value for given inputs were 220 which is 0,1,1 .

10	20	[12, 2, 1, 1, 4, 5, 7, 5, 8, 10]	[4, 2, 1, 2, 10, 15, 3, 10, 4, 8]

max fringe size:

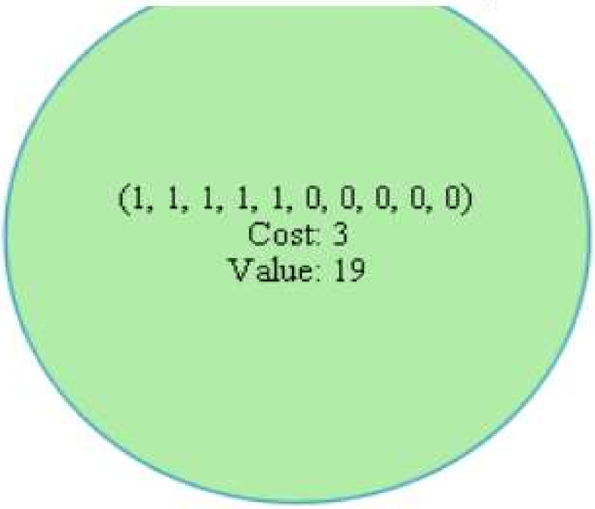
1

visited nodes:

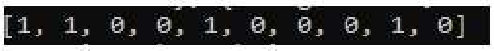
0

iterations:

4



Initial State:

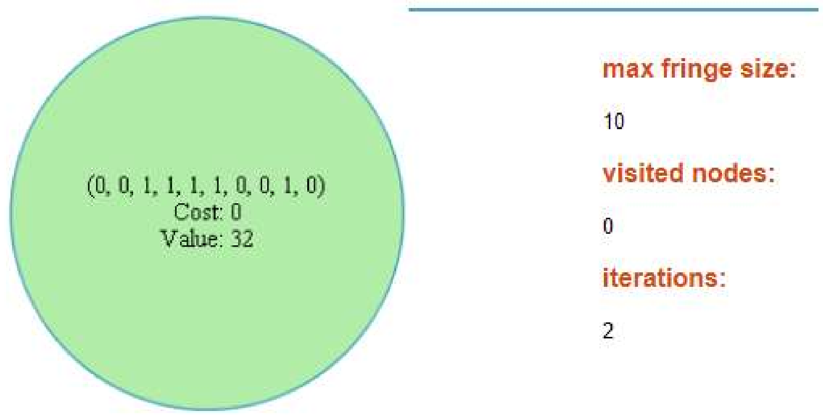


```
[1, 1, 0, 0, 1, 0, 0, 0, 1, 0]
```

The example above hill climbing search did not find global maximum value and found a local maximum. As you can see in other examples (1 ,2 and 3) fringe size did not changed. So we can assume hill climbing search is good at keeping space low.

Genetic Search:

1)



(0, 0, 1, 1, 1, 1, 0, 0, 1, 0)
Cost: 0
Value: 32

max fringe size:

10

visited nodes:

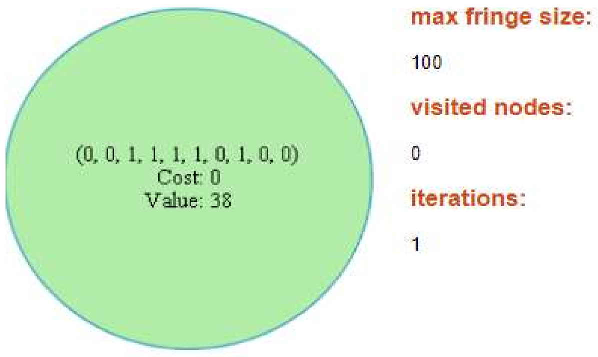
0

iterations:

2

10	20	[12, 2, 1, 1, 4, 5, 7, 5, 8, 10]	[4, 2, 1, 2, 10, 15, 3, 10, 4, 8]

```
result=genetic(problem,population_size=10,viewer=WebViewer())
```



(0, 0, 1, 1, 1, 1, 0, 1, 0, 0)
Cost: 0
Value: 38

max fringe size:

100

visited nodes:

0

iterations:

1

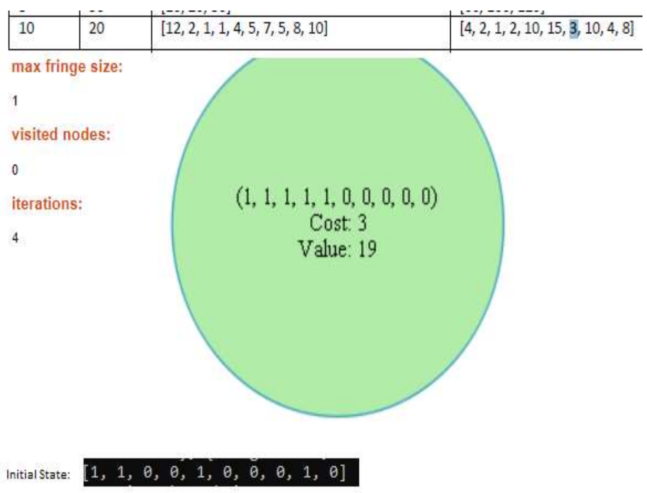
10	20	[12, 2, 1, 1, 4, 5, 7, 5, 8, 10]	[4, 2, 1, 2, 10, 15, 3, 10, 4, 8]

```
result=genetic(problem,population_size=100,viewer=WebViewer())
```

The genetic search works better to find optimal solution when increase the population size but when population size increased required space also gets bigger.

The photo with the population size 10 did not found the optimal solution because there is low chance to find optimal solution sincerely there is low possibility to get optimal state.

Hill Climbing VS Genetic



10	20	[12, 2, 1, 1, 4, 5, 7, 5, 8, 10]	[4, 2, 1, 2, 10, 15, 3, 10, 4, 8]

max fringe size:

1

visited nodes:

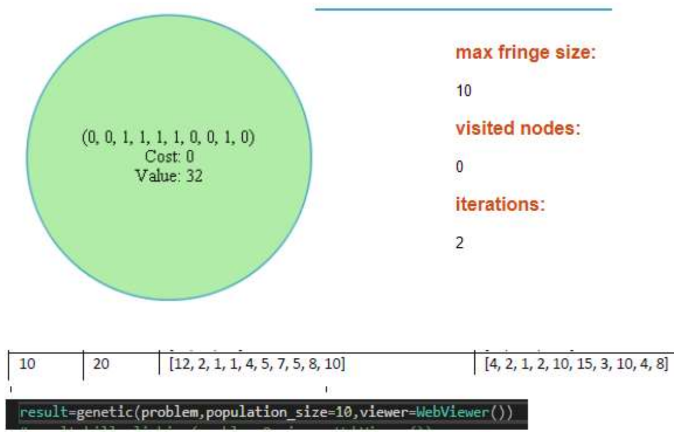
0

iterations:

4

```
(1, 1, 1, 1, 1, 0, 0, 0, 0, 0)
Cost: 3
Value: 19
```

Initial State: `[1, 1, 0, 0, 1, 0, 0, 0, 1, 0]`



```
(0, 0, 1, 1, 1, 1, 0, 0, 1, 0)
Cost: 0
Value: 32
```

max fringe size:

10

visited nodes:

0

iterations:

2

10	20	[12, 2, 1, 1, 4, 5, 7, 5, 8, 10]	[4, 2, 1, 2, 10, 15, 3, 10, 4, 8]

```
result=genetic(problem,population_size=10,viewer=WebViewer())
```

Genetic search is much optimal and faster than hill climbing search when population size is big enough but genetic search requires much more space than hill climbing search. Hill climbing search time complexity is higher than genetic when we look at our iteration counts.