Evolutionary Algorithm for Knapsack Problem

The code implements an evolutionary algorithm to solve the 0-1 knapsack problem. In this problem, each item is assigned a value (1, 2, 3, ...) and a randomly generated weight (between 1 and 10). The goal is to pick a subset of items that maximize total value without exceeding a knapsack capacity (half the total weight of all items). The candidate solution is represented as a bitstring, where each bit indicates whether an item is included.

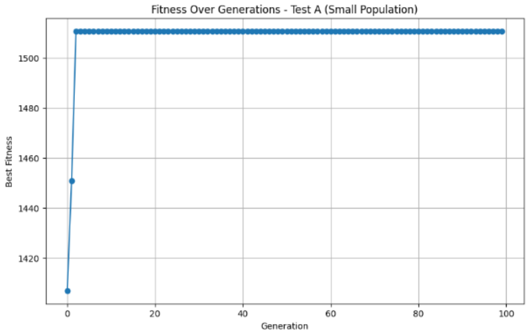
Design Choices and Implementation:

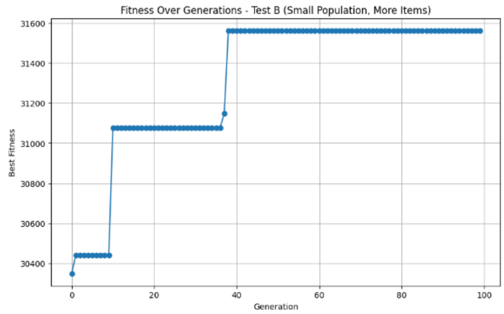
- **Genotype Representation:**
  The candidate solutions are represented as bitstrings of length equal to the number of items. This encoding shows which items to include/exclude for the knapsack.
- **Fitness Function:**
  The fitness function is the sum of values for selected items. If the total weight is within the capacity, the fitness simply equals the total value. If the weight is over the capacity limit, the fitness is penalized by scaling the total value by ${}^{capacity}/_{(totalWeight\ \times 10)}$. The multiplication by 10 was needed to reduce the frequency of infeasible solutions as now the scaling has a bigger impact on the fitness.
- **Crossover and Mutation:**
  A single-point crossover operator recombines parental bitstrings at a randomly chosen point, while mutation randomly flips bits with a fixed probability. These were chosen for simplicity and effectiveness.
- **Selection Strategy:**
  Tournament selection is employed by randomly sampling 40 individuals from the population and choosing the one with the highest fitness. This provides robust selection and preserves diversity.

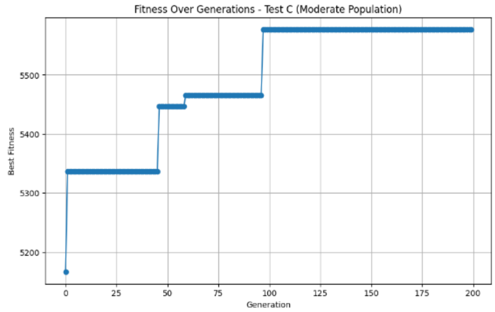
Testing:

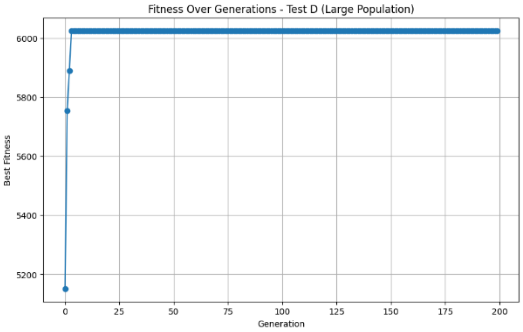
After implementation, I conducted some testing:

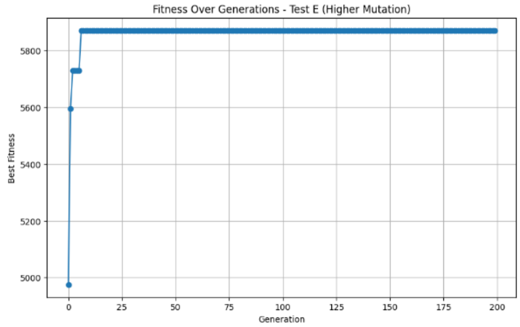
Test	Population	# Items	Generations	Mutation prob	Crossover prob
A	100	20	100	0.1	0.8
B	100	100	100	0.1	0.8
C	500	40	200	0.1	0.8
D	1000	40	200	0.1	0.8
E	500	40	200	0.2	0.4
F	500	40	300	0.4	0.4

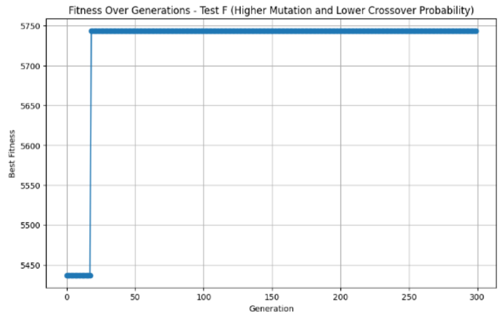












Results and Analysis:

- **Test A:**

```
Knapsack Capacity: 51
Item values: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Item weights: 8 3 5 1 9 2 1 4 4 1 2 9 3 5 1 10 9 9 7 10

Final Best Fitness: 1510.7547169811319
Best Genotype: 00010111010111001000
Total Value: 91.0
Total Weight: 35.0 (Feasible)
```

  Convergence was very fast, with the best fitness stabilizing after just a few generations. The final solution was feasible with a total value of 91 and a weight of 35.

- **Test B:**

```
Knapsack Capacity: 241
Item values: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
Item weights: 3 3 5 3 1 5 8 4 5 1 8 4 5 5 1 9 4 4 3 7 3 9 4 5 9 9 10 5 5 6 6 7 10 2 3 2 7 10 7 4 6 4 5 5 1 8 2 5 8 9 3 5 1 6 9 2 1 3 6 8 2 8 10 1 7 5 3 4 3 4 5 3 8 1 5 3 5 3 6 4 4 3 1 2 6 7 1 4 6 1 7 3 8 5 2 6 3 8 3 4

Final Best Fitness: 31563.225806451617
Best Genotype: 0000111010011111010001001100010010010100011111011011101100110011000001000001000001001111010001000001
Total Value: 2013.0
Total Weight: 221.0 (Feasible)
```

  With more items, we see how the algorithm is making step improvements, eventually finding the optimal solution with high fitness.

- **Test C:**

```
Knapsack Capacity: 100
Item values: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Item weights: 5 3 4 9 2 9 9 2 1 2 1 4 5 6 7 5 7 6 2 5 8 9 3 5 3 7 5 9 3 4 2 7 1 7 1 8 8 9 4 4

Final Best Fitness: 5576.923076923077
Best Genotype: 0100111110100100001010000101010000100000
Total Value: 221.0
Total Weight: 64.0 (Feasible)
```

```
Knapsack Capacity: 106
Item values: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Item weights: 3 9 9 8 4 6 5 7 9 8 9 7 4 3 2 1 1 3 1 9 6 5 10 5 4 3 7 2 9 7 1 8 4 9 7 1 3 7 5 1

Final Best Fitness: 6026.2962962962965
Best Genotype: 0111010000001101111000010100010110010110
Total Value: 370.0
Total Weight: 85.0 (Feasible)
```

  Increasing the population size and generations produced better solutions. Test D (large population) achieved a final best fitness of approximately 6026, with a total value of 370 and weight of 85.

- **Test E:**

```
Knapsack Capacity: 108
Item values: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
Item weights: 6 9 10 10 8 7 2 8 6 8 9 1 7 2 4 2 8 1 8 6 6 7 3 2 10 3 1 6 4 4 5 9 5 6 3 2 6 6 4 2

Final Best Fitness: 5871.272727272727
Best Genotype: 1100010110011100100001011111101110000000
Total Value: 359.0
Total Weight: 106.0 (Feasible)
```

With a higher mutation rate, the final best fitness was slightly lower. This suggests that while higher mutation can improve exploration, it may as well disrupt convergence.