Scott Bebington u21546216

Assignment 2

Contents

1. GA Configuration Description

For the Genetic Algorithm (GA), I decided upon the following parameters:

Population Size: Through trial and error, I found a population size of around 100 for each knapsack, this was to ensure there was a good variety of choice when testing combinations i.e. A large search space and good diversity.

Crossover: I set a randomised value between 1 and the size of the chromosome to determine the crossover point.

Selection technique: I used a tournament style crossover technique ensuring a balance between exploration and exploitation.

Mutation Rate: The mutation rate was set to only effect 1 in 1000 chromosomes, controlling the probability of random mutation in offspring chromosomes.

Termination Criteria: The algorithm terminates after 1000 generations or if the known optimum is reached.

Steps:

1. Start with a population size of 100 randomly generated chromosomes representing whether the item is in the bag or not (e.g. 01001101).
2. Evaluate the fitness of each of the chromosomes, if it exceeds the weight limit the fitness is set to 0, otherwise the fitness is the total value of the knapsack.
3. Select new parents for the new generation.
    a. If the total fitness is 0, it means that the weight limit for every individual exceeded the maximum weight limit, In this case, the function selects parents randomly. It goes over the population and for each individual, it generates a random index and selects the individual at that index to be a parent. Additionally, it introduces a few new individuals with all 0 chromosomes to decrease the items in the knapsack during crossover.
    b. If the total fitness score is not 0 then some individuals are within the weight limit. In this case the parents are selected based on their fitness score.
4. Take each set of parents and apply the crossover to create the new generation by splitting each parents chromosomes by the randomly generated crossover point.
5. Make the new generation the sample population.
6. Introduce and mutation that effects 1 in 1000 individuals, this is to ensure that the gene pool is further diversified.
7. The best fitness score is then taken and tested against the known optimum, if they are the same then the algorithm terminates, otherwise repeat steps 2 to 7 until 1000 generations have completed.

2. GA + Local search Configuration Description

This algorithm uses the same parameters as just the GA

Steps:

1. Start with a population size of 100 randomly generated chromosomes representing whether the item is in the bag or not (e.g. 01001101).
2. Apply the Local Search
    a. For each chromosome, apply a local search for gene.
    b. If the result is better than the original, mark the improvement and move onto the next chromosome.
    c. If the result is worse, revert the gene back to its original state and move onto the next one.
    d. Complete the algorithm when no further improvements have been made
3. Evaluate the fitness of each of the chromosomes, if it exceeds the weight limit the fitness is set to 0, otherwise the fitness is the total value of the knapsack.
4. Select new parents for the new generation.
    a. If the total fitness is 0, it means that the weight limit for every individual exceeded the maximum weight limit, In this case, the function selects parents randomly. It goes over the population and for each individual, it generates a random index and selects the individual at that index to be a parent. Additionally, it introduces a few new individuals with all 0 chromosomes to decrease the items in the knapsack during crossover.
    b. If the total fitness score is not 0 then some individuals are within the weight limit. In this case the parents are selected based on their fitness score.
5. Take each set of parents and apply the crossover to create the new generation by splitting each parents chromosomes by the randomly generated crossover point.
6. Make the new generation the sample population.
7. Introduce and mutation that effects 1 in 1000 individuals, this is to ensure that the gene pool is further diversified.
8. The best fitness score is then taken and tested against the known optimum, if they are the same then the algorithm terminates, otherwise repeat steps 2 to 7 until 1000 generations have completed.

3. Description and Justification of Local Search

Hill climbing was chosen as the local search technique due to its simple nature, ease of implementation and effectiveness in improving solutions within a local neighbourhood.

The reason behind choosing the hill climb algorithm over everything else is its compatibility with the Genetic Algorithm. Since the problem revolves around selecting items iteratively, I believe it was perfect for this problem.

4. Experimental setup

Problem Instances:

I made use of the given problem instances provided with known optima for each one. These instances were selected to represent a range of scenarios and sizes commonly encountered in the knapsack problem.

Parameter Settings for Genetic Algorithm (GA) and GA + Local Search:

The genetic algorithm was configured with specific parameters to fine tune the problem solving process. These parameters were chosen based trial and error to provide the best results with the shortest runtime.

Population Size: 100

Mutation Rate: 1 in 1000

Selection Mechanism: Tournament style selection

Maximum Generation: 1000

Termination Criteria: Maximum number of generations reached or known optimum reached.

Local Search choice (GA +LS): Hill climb

Seed Value: for each problem instance a specific seed value was chosen to give consistent and good results. Please see Table of results for seed values.

PC specifications:
Processor: AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz

RAM: 16,0 GB

Operating system: Windows 11 running wsl for this assignment.

5. Table of results

These results were obtained using the default seed values when no specific value is selected

Problem Instance	Algorithm	Seed Value	Best Solution	Known Optimum	Runtime (Seconds)
f1_l-d_kp_10_269	GA	1714247743	295	295	0.00105
f1_l-d_kp_10_269	GA	1714247743	295	295	0.001819
f2_l-d_kp_20_878	GA	1714241775	1024	1024	0.018734
f2_l-d_kp_20_878	GA	1714241775	1024	1024	0.158905
f3_l-d_kp_4_20	GA	1714241775	35	35	0.000474
f3_l-d_kp_4_20	GA	1714241775	35	35	0.000859
f4_l-d_kp_4_11	GA	1714241775	23	23	0.000478
f4_l-d_kp_4_11	GA	1714241775	23	23	0.000869
f5_l-d_kp_15_375	GA	1714245778	481.069	481.069	0.418058
f5_l-d_kp_15_375	GA	1714245778	481.069	481.069	2.42552
f6_l-d_kp_10_60	GA	1714241775	52	52	0.000638
f6_l-d_kp_10_60	GA	1714241775	52	52	0.00185
f7_l-d_kp_7_50	GA	1714302749	107	107	0.000558
f7_l-d_kp_7_50	GA	1714302749	107	107	0.002648
f8_l-d_kp_23_10000	GA	1714302872	9736	9767	0.505036
f8_l-d_kp_23_10000	GA	1714302872	9576	9767	3.70489
f9_l-d_kp_5_80	GA	1714241775	130	130	0.000519
f9_l-d_kp_5_80	GA	1714241775	130	130	0.001032
f10_l-d_kp_20_879	GA	1714245778	1025	1025	0.060086
f10_l-d_kp_20_879	GA	1714245778	1025	1025	0.097781
knapPI_1_100_1000_1	GA	1714241776	9147	9147	0.876457
knapPI_1_100_1000_1	GA	1714241776	9147	9147	12.8242

6. Statistical analysis

Hypotheses:

Null Hypothesis (H0): µGA = µGA+LS

Alternative Hypothesis (H1): µGA != µGA+LS

Using the Best solution values side by side to perform the z-test with an alpha value of 0.05 we get the following results.

	GA	GA + LS
	295	295
	1024	1024
	35	35
	23	23
	477	477
	52	52
	107	107
	9736	9576
	130	130
	1025	1025
	9147	9147
Variances	13669963	13424887

z-Test: Two Sample for Means

	GA	*GA + LS*
Mean	2004,636364	1990,090909
Known Variance	13669963	13424887
Observations	11	11
Hypothesized Mean Difference	0	
z	0,009267877	
P(Z<=z) one-tail	0,496302705	
z Critical one-tail	1,644853627	
P(Z<=z) two-tail	0,99260541	
z Critical two-tail	1,959963985	

Using these results and taking into consideration that it is a 1-sided test. The p value for a 1-sided test is bigger than 0.05 and therefor we can accept the null hypothesis(H0) that the 2 means are equivalent

7. Critical analysis

Although the use of a local search is intended to speed up the search process by changing values for the better. In the case of the knapsack problem using the Genetic Algorithm, The GA + LS algorithm was in every case slower in terms of runtime by almost double and on certain occasions even provided worse results.

In the interest of finding the best solution with the fastest time I would recommend just using the Genetic Algorithm rather than both.