Genetic Algorithm Configuration Description

Parameter	Value
Population Size	100
Selection Method	Tournament
Crossover rate	90%
Mutation rate	1%
Mutation type	Bit-flip
Fitness Function	Value of the items
Maximum Generations	100

1. **Knapsack Problem Definition**

- **Objective**: Maximize total value of selected items within knapsack capacity limit.
- **Constraints**: Total weight of selected items must not exceed knapsack capacity.
- **Items**: Each item has a weight and value.

2. **Genetic Algorithm (GA) Configuration**

- **Chromosome Representation**: Each chromosome is a binary string where each gene represents whether an item has been selected (1) or excluded (0) from the knapsack.
- **Fitness Function**: The fitness of a chromosome is the total value of items included in the knapsack, given that the total weight does not exceed capacity. Otherwise, the fitness of that chromosome is assigned a penalty score of 0.
- **Selection Method**: Tournament selection is used to choose parent chromosomes for reproduction.
- **Genetic Operators**:
    - **Crossover**: Single-point crossover is performed at a rate of 0.9. A rate of 0.9 is chosen in order to allow for more exploitation in the search space.
    - **Mutation**: Bit-flip mutation occurs at a rate of 0.01, ensuring no duplication of items and more consistency in results.
    - **Note**: **Parameters come from reference [1]**

Genetic Algorithm with Local Search Configuration and Description

Parameter	Value
Population Size	100
Selection Method	Tournament
Crossover rate	90%
Mutation rate	1%
Mutation type	Bit-flip
Fitness Function	Value of the items
Maximum Generations	100

1. **GA + Local Search (ILS) Configuration**:
    - **GA Configuration**:
        - **Chromosome Representation**: Each chromosome is a binary string where each gene represents whether an item has been selected (1) or excluded (0) from the knapsack.
        - **Fitness Function**: The fitness of a chromosome is the total value of items included in the knapsack, given that the total weight does not exceed capacity. Otherwise, the fitness of that chromosome is assigned a penalty score of 0.
        - **Selection Method**: Tournament selection is used to choose parent chromosomes for reproduction.
        - **Genetic Operators**:
            - Crossover: Single-point crossover is performed at a rate of 0.9. A rate of 0.9 is chosen in order to allow for more exploitation in the search space.
            - Mutation: Bit-flip mutation occurs at a rate of 0.01, ensuring no duplication of items and more consistency in results.
        - **Note**: **Parameters come from reference [1]**

- **Iterated Local Search Integration (ILS)**:
    - After GA operations, apply ILS to improve solutions iteratively.
    - ILS explores the neighbourhood of existing solutions by perturbing and applying local search.
    - **Justification**:
        - **ILS Exploration**: ILS perturbs solutions, exploring different neighbourhoods.
        - **Local Improvement**: Local search within ILS focuses on improving the current solution.
        - **Escape Local Optima**: ILS can escape local optima by introducing randomness.
        - **Complement to GA**: ILS complements GA's global search by refining solutions locally.

2. **Iterated Local Search (ILS) Method**:
    - **Method**: **Perturbation + Local Search**:
        - **Perturbation**:
            - Randomly swap two items in the solution to create a perturbed solution.
            - Introduces diversity and explores a different region of the search space.
        - **Local Search**:
            - Apply local search (similar to hill climbing) to the perturbed solution.
            - Explore neighbouring solutions and improve fitness.
            - Repeat until no further improvement is possible.
        - **Iteration**:
            - Repeat perturbation and local search for a fixed number of iterations.

A table presenting the results.

Problem Instance	Algorithm	Seed Value	Best Solution	Known Optimum	Runtime (seconds)
f1_l-d_kp_10_269	GA-LS GA	1714210631751 1714209323205	295 295	295 295	0.012 0.007
f2_l-d_kp_20_878	GA-LS GA	1714210698173 1713877276773	1024 1024	1024 1024	0.017 0.006
f3_l-d_kp_4_20	GA-LS GA	1714210729761 1713873038380	35 35	35 35	0.004 0.004
f4_l-d_kp_4_11	GA-LS GA	1714210761531 1713873102948	23 23	23 23	0.005 0.006
f5_l-d_kp_15_375	GA-LS GA	1714210802747 1713875124154	481,0694 481,0694	481,0694 481,0694	0.01 0.007
f6_l-d_kp_10_60	GA-LS GA	1714210841104 1714209831671	52 52	52 52	0.009 0.005
f7_l-d_kp_7_50	GA-LS GA	1714210890982 1714209915358	107 107	107 107	0.006 0.006
f8_l-d_kp_23_10000	GA-LS GA	1714210959527 1714209990033	9767 9752	9767 9767	0.024 0.006
f9_l-d_kp_5_80	GA-LS GA	1714210994357 1713875416342	130 130	130 130	0.006 0.004
f10_l-d_kp_20_879	GA-LS GA	1714211048777 1714210166834	1025 1019	1025 1025	0.015 0.007
knapPI_1_100_1000_1	GA-LS GA	1714207477702 1714207314650	6295 4491	9147 9147	0.135 0.017

Statistical Analysis

Mean performance of GA-LS ($\bar{x}_1$): 0.02209

Standard deviation of GA-LS (σ_1): 0.03616

Mean performance of GA ($\bar{x}_2$): 0.00681

Standard deviation of GA (σ_2): 0.00338

$$z = \frac{(\bar{x}_1 - \bar{x}_2)}{\sqrt{\frac{\sigma^2}{n_1} + \frac{\sigma^2}{n_2}}}$$

$$z = \frac{(0.02209 - 0.00681)}{\sqrt{\frac{0.03616^2}{11} + \frac{0.00338^2}{11}}}$$

$$z = 1.395$$

A one-tailed z-test at the 5% level has a critical z-value of 1.695. Since 1.395 < 1.695, we do not have sufficient evidence to reject the null hypothesis. Therefore, there is no significant difference in performance between a genetic algorithm and a genetic algorithm with a local search.

Analysis of the Results

From the results above and when testing each programme, one can conclude that a Genetic Algorithm with a Local Search (GA-LS) is more accurate across multiple programme runs and is more likely to find the known optimums. Furthermore, it can be noted that in most cases the GA-LS was slower in terms of performance but as seen in the statistical analysis above the difference in performance is negligible at a 5% level. The reason for the GA-LS being able to more accurately determine the known optimum is due to the fact that the Iterated Local Search (ILS), being used with the Genetic Algorithm, looks to iteratively improve upon solutions. It is important to note however that for one problem instance neither programme was able to find the known optimum. This possibly could have been improved by tuning the parameters differently or by making an improvement to the genetic algorithm itself and how possible solutions are created.

References for Parameter tuning

Tabassum, M. and Mathew, K., 2014. A genetic algorithm analysis towards optimization solutions. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, *4*(1), pp.124-142. [1]