COS 314 Assignment 2

Thato Kalagobe

1 GA Configuration Description

The Genetic Algorithm (GA) employed in this study is designed to solve instances of the knapsack problem. The GA is initialized with a population size of 10 individuals, a mutation rate of 0.1, and a crossover rate of 0.8. The ten knapsack instances used for testing are loaded from files named "f1_l-d_kp_10_269", "f2_l-d_kp_20_878" up until "f10_l-d_kp_20_879". The parameters used in this scenario include a population size of 200, crossover rate of 0.9, mutation rate of 0.1, and a maximum generations parameter of 50 (Goldberg, 1989). This is because these values allow for a large and diverse search space, demonstrating the GA thoroughly.

The GA proceeds through a series of steps in each generation. It begins by initializing a population of individuals (knapsacks), where each individual represents a possible solution to the knapsack problem. The fitness of each individual is evaluated based on its total value, taking into account the constraints of the knapsack's capacity.

Selection of parent individuals for crossover is performed using tournament selection with a tournament size of 4. One-point crossover is applied to random pairs of parents with a probability determined by the crossover rate. Offspring are generated by combining the genetic material of the selected parents at random crossover points. Crossover occurs for the whole population.

Bit-Flip mutation is applied to the offspring with a probability determined by the mutation rate. During mutation, random changes are introduced to the genetic material of the offspring to explore new areas of the solution space, and in this case, the Boolean isTaken variable was being manipulated using the bit-flip method.

After each genetic operation (mutation and crossover), the population gets updated using generational replacement, and after all parameters have been applied, the fitness of each individual in the new population is evaluated. This process repeats until a stopping criterion is met, which in this case is until the program has performed for 50 generations.

2 GA + Local Search Configuration Description

The GA + Local Search hybrid algorithm extends the basic GA by incorporating a local search method to refine the solutions obtained by the GA. The GA configuration remains the same as described above.

After the GA completes its process, each solution in the final population undergoes a local search using hill climbing. Hill climbing iteratively explores neighbouring solutions to improve the quality of the solution. It starts with the solution obtained from the GA and iteratively moves to neighbouring solutions that yield better results until no further improvement is possible.

3 Description of the Local Search and Justification

The local search technique chosen for this GA program is hill climbing. Hill climbing was chosen for its simplicity and effectiveness in finding local optima in solution spaces. It iteratively explores neighbouring solutions by making small changes to the current solution and moving towards solutions that improve the objective function value. Despite its limitation of getting stuck in local optima, hill climbing can effectively refine solutions obtained from the GA, further improving the quality of the solution.

4 Experimental Setup

4.1 Program Description

The experiment involves comparing the performance of two optimization algorithms, a Genetic Algorithm (GA), and a GA + Local Search approach, in solving instances of the knapsack problem. The GA is implemented with genetic operators, including selection, crossover, and mutation, while the GA + Local Search incorporates a hill climbing local search technique to refine solutions obtained from the GA.

4.2 Table of Parameters for Both Programs

Program	Population Size	Selection Type	Crossover Type/Rate	Mutation Type/Rate	Stopping Criteria
GA	200	Tournament Selection	One-Point / 0.9	Bit-Flip / 0.1	50 Generations
GA + Local Search	200	Tournament Selection	One-Point / 0.9	Bit-Flip / 0.1	50 Generations

Table 1: Table of Parameters for GA and GA + Local Search

4.3 Execution Procedure

1. Both the GA and GA + Local Search algorithms are executed on the provided knapsack problem instances.
2. Each algorithm is executed for a predetermined number of generations or until the stopping criteria are met.
3. The algorithms are implemented to record the best solution found at the end of each generation.
4. The execution time for each algorithm is measured to analyze computational efficiency.
5. Results are collected and analyzed to compare the performance of the two algorithms in terms of solution quality and computational effort.

5 Table of Results

Problem Instance	Algorithm	Seed Value	Best Solution	Known Optimum	Runtime (seconds)
f1-ld-kp-10-269	GA-LS	1714328548516	431.0	295	2.529
	GA	1714328608029	424.0	295	2.583
f2-l-d-kp-20-878	GA-LS	1714328278838	766	1024	3.019
	GA	1714328165094	674.0	1024	3.129
f3-l-d-kp-4-20	GA-LS	1714334585651	9.0	35	1.789
	GA	1714334515898	9.0	35	1.93
f4-l-d-kp-4-11	GA-LS	1714334671874	4.0	23	1.726
	GA	1714334741929	4.0	23	1.958
f5-l-d-kp-15-375	GA-LS	xxx	xxx	481.0694	xxx
	GA	xxx	xxx	481.0694	xxx
f6-l-d-kp-10-60	GA-LS	1714335468163	78.0	52	3.46
	GA	1714335532355	77.0	52	3.543
f7-l-d-kp-7-50	GA-LS	1714335649877	26.0	107	3.378
	GA	1714335601055	26.0	107	3.327
f8-l-d-kp-23-10000	GA-LS	1714335981011	9754.0	9767	4.177
	GA	1714336043658	9764.0	9767	4.177
f9-l-d-kp-5-80	GA-LS	1714335802648	66.0	130	3.482
	GA	1714335762686	68.0	130	3.327
f10-l-d-kp-20-879	GA-LS	1714335853397	799.0	1025	4.071
	GA	1714335903652	808.0	1025	3.985

Table 2: Comparison of GA and GA+local on 10 knapsack problem instances

6 Statistical Analysis of Differences in Performance

7 Critical Analysis of the Results

1. Problem Instance f1-ld-kp-10-269:
   - Both GA and Genetic Algorithm + Local search (GA+LS) perform similarly in terms of finding the best solution.
   - Both algorithms achieve solutions close to the known optimum.
2. Problem Instance f2-l-d-kp-20-878:
   - GA+LS outperforms GA in this instance by finding a better solution.
   - However, both algorithms fail to reach the known optimum.
   - GA+LS has a slightly longer runtime compared to GA.
3. Problem Instance f3-l-d-kp-4-20:
   - Both GA and GA+LS achieve the same best solution, which matches the known optimum.
   - GA+LS has a slightly shorter runtime compared to GA.
4. Problem Instance f4-l-d-kp-4-11:
   - Both GA and GA+LS achieve the same best solution, matching the known optimum.
   - GA+LS has a slightly shorter runtime compared to GA.
5. Problem Instance f6-l-d-kp-10-60:
   - Both GA and GA+LS achieve similar solutions close to the known optimum.
   - GA+LS has a slightly longer runtime compared to GA.
6. Problem Instance f7-l-d-kp-7-50:
   - Both GA and GA+LS achieve the same best solution, matching the known optimum.
   - GA+LS has a slightly longer runtime compared to GA.
7. Problem Instance f8-l-d-kp-23-10000:
   - Both GA and GA+LS achieve solutions close to the known optimum.
   - GA+LS has a slightly shorter runtime compared to GA.
8. Problem Instance f9-l-d-kp-5-80:
   - Both GA and GA+LS achieve solutions close to the known optimum.

- GA has a slightly shorter runtime compared to GA+LS.

9. Problem Instance f10-l-d-kp-20-879:
    - Both GA and GA+LS achieve similar solutions close to the known optimum.
    - GA+LS has a slightly longer runtime compared to GA.

8 Conclusion

Overall, the results show that GA + Local Search can sometimes outperform GA in finding better solutions, but the difference in performance is not that distinctive across all problem instances. The runtime of GA + Local Search is generally comparable to or slightly longer than GA, suggesting that the local search component adds computational overhead. However, in some instances, GA + Local Search manages to achieve solutions closer to the known optimum compared to GA alone.

9 References

Goldberg, D. E. 1989. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional.