Paul Pilane | u21448150

COS 314 | Artificial Intelligence

Assignment 2

Introduction

The knapsack problem is a classic optimization problem in computer science and artificial intelligence. It involves finding the best way to pack a knapsack with a set of items, each with a weight and a value, such that the total weight of the items in the knapsack is below a certain threshold (the capacity of the knapsack), and the total value of the items in the knapsack is maximized. To implement, or solve, this optimization problem, two metaheuristic algorithms are used, namely: (1) Ant Colony Optimization and (2) Genetic Algorithms. The respective algorithms will also be compared, with each other, to analyse their performance at solving the classic optimization at hand.

Ant Colony Optimization

Configuration Description

Ant Decision Rule

Exploitation and exploration strategies used in the ACO implementation. An ant selects the item with the highest pheromone level (exploitation) or select an item probabilistically based on the pheromone level (exploration). The choice of strategy depends on a random number r, compared with a fixed threshold Q0. If r is less than Q0, the ant selects the item with the highest pheromone level if its weight doesn't exceed the knapsack's capacity, otherwise, it selects no item. If r is greater than or equal to Q0, the ant selects an item based on a probability proportional to its pheromone level if its weight doesn't exceed the knapsack's capacity, otherwise, it selects no item.

Local Search Strategy

The local search used is the Greedy Hill Climbing. The maximum number of iterations is the stopping criterion. The maximum number of iterations is 50. A

Termination Criteria

The termination criterion is when the ants have explored after MAX_ITERATIONS number of times, such that they have better chances for finding the better solutions. The MAX_ITERATIONS is 500 which enable ants to sufficiently explore and exploit in other potential better places. In essence, to find better solutions.

Heuristic Information

The heuristic information is value of the pheromone trails associated with each item in the knapsack. The pheromone levels of each item are used to calculate the probability of selecting the item during the construction of a solution. The probability calculation is based on the ratio of the pheromone level of the item in the 0 state to the sum of the pheromone levels of the item in the 0 and 1 states. This heuristic information considers the pheromone level and biases the selection of items towards the ones with higher pheromone levels. The pheromone levels are associated with each state of each item in the knapsack, representing the attractiveness or quality of that state

Experimental Setup

Table of Parameters

Number of Ants	Maximum number of iterations	ALPHA	BETA	EVAPORATION_RATE	INITIAL_PHEROMONE
100	500	1.0	5.0	0.5	1.0

Problem Representation

The pheromone trail is a matrix, which represents a graph by which the artificial ants traverse to find better solutions. Therefore, at each node of the graph, there exists a candidate solution to the Knapsack Problem. The candidate solution is a string of binary values that indicate which items have been selected (1) and not selected (0), to be placed inside the knapsack. By traversing the graph and updating the pheromone trail, the ants can effectively search for high-quality solutions to the Knapsack Problem.

the ones with higher pheromone levels. The pheromone levels are associated with each state of each item in the knapsack, representing the attractiveness or quality of that state

Fitness function

The function calculates the total value and weight of the included items and returns the total value if the total weight is within the capacity of the knapsack, otherwise it returns -1 to indicate an infeasible solution.

Genetic Algorithm

Experimental Setup

Problem Representation

Each item to be placed in the knapsack is represented by a single gene. The binary value of the gene indicates whether the corresponding item is selected or not. Therefore, the collective set of genes represents a chromosome that encodes a candidate solution to the Knapsack Problem, where each gene corresponds to an item and its binary value indicates whether the item is included in the knapsack or not.

that can help to escape from local optima and explore new regions of the search space. The bit-flip increase the diversity of the population and prevent premature convergence.

Fitness Function

The fitness function calculates the total value and weight of the items in the knapsack for an individual, applies a penalty factor based on the knapsack's capacity, and returns the fitness score as the product of the total value and penalty factor. The fitness score is used to guide the search towards better solutions that maximize the total value while respecting the knapsack's capacity.

Configuration Description

Table of Parameters

POPULATION_SIZE	MAX_GENERATIONS	MUTATION_RATE	CROSSSOVER_RATE
100	12	0.03	0.6

Crossover Operator

The crossover operator is One Point Crossover. In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs, as seen in [1]. A higher cross over rate is preferrable, such can be understood in this following paper [2]

Selection Type

The selection type for parent selection is the Tournament-Selection. Such a selection type can help preserve diversity in the population, as it allows individuals with different fitness levels to compete against each other. This can prevent premature convergence, which occurs when the population converges to a suboptimal solution too quickly, such that it does not really have the best solution.

Mutation Operator

The mutation Operator implemented is a bit-flip mutation operator. The mutation operator randomly selects a gene (bit) in the chromosome (binary string) and flips its value. elects a random gene (bit) in the genes list and flips its value from 0 to 1 or from 1 to 0 by subtracting the current value from 1. The effect of this operation is to introduce a small random change in the chromosome such that enables the algorithm to escape the local opima, if stuck on one, to find better solutions.

The mutation rate should be disproportionate to the crossover rate. According to [2], the higher the mutation rate, the more harmful it is to the GA optimal performance. Thus, the combination of a low mutation rate and higher cross over rate gives optimal results

Termination Criteria

The termination criterion is when the ants have explored after MAX_GENERATIONS number of times, such that they have better chances for finding the better solutions. The MAX_ GENERATIONS is 12. This is in conjunction with the higher number of population which will ensure a potential best solution

Results Table

Problem Instance	Algorithm	Best Solution	Known Optimum	Runtime (seconds)
f1_l-d_kp_10_269	ACO	283.0	295	0.36
	GA	293.96	295	0.74
f2_l-d_kp_20_878	ACO	418.0	1024	1.28
	GA	1024.66	1024	0.96
f3_l-d_kp_4_20	ACO	35.0	35	0.16
	GA	30.83	35	0.49
f4_l-d_kp_4_11	ACO	23.0	23	0.07
	GA	17.15	23	0.54
f5_l-d_kp_15_375	ACO	296.676534	481.0694	0.63
	GA	481.07	481.0694	1.21
f6_l-d_kp_10_60	ACO	52	52	0.17
	GA	50.08	52	0.72
f7_l-d_kp_7_50	ACO	107	107	0.11
	GA	96.35	107	0.74
f8_l-d_kp_23_10000	ACO	6182.0	9767	1.06
	GA	9767.0	9767	1.97
f9_l-d_kp_5_80	ACO	130.00	130	0.081
	GA	130.00	130	0.73
f10_l-d_kp_20_879	ACO	518	1025.00	1.99
	GA	1025.00	1025.00	1.23

Statistical Analysis using the Ranking Method based on runtimes

Key points:

1. Faster runtime
2. Slower runtime

Problem Instance		
f1_l-d_kp_10_269	ACO	1
	GA	2
f2_l-d_kp_20_878	ACO	2
	GA	1
f3_l-d_kp_4_20	ACO	1
	GA	2

f4_l-d_kp_4_11	ACO	1
	GA	2
f5_l-d_kp_15_375	ACO	1
	GA	2
f6_l-d_kp_10_60	ACO	1
	GA	2
f7_l-d_kp_7_50	ACO	1
	GA	2
f8_l-d_kp_23_10000	ACO	1
	GA	2
f9_l-d_kp_5_80	ACO	1
	GA	2
f10_l-d_kp_20_879	ACO	2
	GA	1
knapPI_1_100_1000_1	ACO	2
	GA	1

Critical Analysis

According to the results of the experiment, the Ant Colony Optimization algorithm (ACO) outperforms the Genetic Algorithm (GA) in terms of runtime. However, when considering the effectiveness of the solutions and their proximity to the optimal solution, the GA produces more effective results

The reason for such occurrence could be several reasons:

Parameter tuning: The performance of both algorithms is heavily dependent on the choice of parameters such as mutation rate, crossover rate, pheromone update rate, and population size. The parameters became more suitable for the ACO than GA, as they were fine-tuned over time, and ultimately ACO was able to outperform GA.

Convergence rate: Amidst the running of the GA, the GA potentially converged prematurely, where the algorithm gets stuck at a local optimum. ACO, on the other hand, has a slower convergence rate, but is less prone to getting stuck at a local optimum. This could mean that ACO was able to continue improving its solutions over time while GA had already converged to a suboptimal solution.

References

[1] TutorialPoiint, “Genetic Algorithms - Crossover,” [Online]. Available: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm. [Accessed 17 May 2023].

[2] P. V. a. P. D.D, “THE OPTIMAL CROSSOVER OR MUTATION RATES IN GENETIC,” 2014. [Online]. Available: https://www.cibtech.org/J-ENGINEERING-TECHNOLOGY/PUBLICATIONS/2015/VOL-5-NO-3/05-JET-006-PATIL-MUTATION.pdf. [Accessed 17 May 2023].