KNAPSACK (ACO)

Hector Manuel Martinez Jiménez

September 2023

1 Introduction

The objective of this partial evaluation work is to solve the following problem: A thief arrives to rob a house. Luckily, he finds a room filled with extremely valuable objects. Unluckily, he only has a small backpack that can carry a maximum of 10 kilograms. Therefore, he must decide which objects to take, trying to maximize his profit. Among the objects he found are the following:

	Centenaries bag	**Wad of $1000 bills**	**Large jewelry box**	**Small jewelry box**	**Stamp collection**	**Art**	**Gold paperweight**
Profit	$750,000	$500,000	$2,750,000	$950,000	$1,850,000	$3,250,000	$3,950,000
Weight	2.5 kg.	1 kg	6 kg	2.5 kg	1.5 kg	3 kg	5kg

The formulation of the above problem is nothing less than a version of the well-known knapsack problem.

2 Solution

To solve the knapsack problem, we will use a well-known algorithm in bioinspired algorithms, ant colony optimization (ACO). Although in the implementation of ACO, a distance matrix is typically used (representing the distance between each node of the graph defined for the problem), in this solution, we will not work with matrices but with independent lists: one for the profit of the elements, another for the weight, and another for the pheromone level of each object. It's important to note that while the lists for each set of data are independent, we simulate the behavior of a matrix because the index of each list corresponds to the information of the same object.
With this situation in mind, the next step is to discuss the procedure to solve our problem with the structures outlined earlier. The implementation of our code is based on the following pseudocode:

begin
while (a cycle exists) **do**
while (an ant k, which has not yet worked, exists) **do**
while ($V_c > 0$) **do**
select a next object o_j with probability p_j
add a selected object to a partial solution $S = S + \{o_j\}$
update the current knapsack load capacity $V_C = V_C - w_j$
update the profit $Z = Z + z_j$
update neighborhood of the current state $N_i = \{o_i : w_i \leq V_C\}$
end
remember the best solution if a better solution has been found
end
remember a global best solution if a better solution has been found
use an evaporation mechanism $T = \rho T$
update pheromone trails $T = T + \Delta T$
end.
This pseudocode is based on the following formula, where each parameter is defined as follows:

$$p_j = \frac{\tau_j^\alpha \eta_j^\beta}{\sum_{j \subseteq N_i} \tau_j^\alpha \eta_j^\beta}$$

j = State
N = Number of items
p_j = Transition probability
τ_j^α = Pheromone track
η_j^β = Attractiveness of the move

After conducting research on the background of this problem, we found a Python implementation that solves our problem well. We will use it as a base for our work.
Modifications made to the base algorithm include:
- Data reading: We changed the way information was entered and accessed within the program because the original implementation did not allow us to use our information optimally.
- Adjustment of suitable parameters for optimal solution.
- Code documentation.
- Rewriting specific code that was poorly implemented, not optimal, and confusing.

3 Results and Analysis

The knapsack problem posed is extremely small, with few options and, therefore, few paths to choose from. As a result, the implementation of this code provides correct results from the initial iterations, even with a small number of particles.
The optimal solution to our problem can be obtained efficiently using the following parameters:

$$\begin{gathered}\text{alpha} = 1 \\ \text{beta} = 0.02 \\ \text{rho} = 0.05 \\ \text{ant-count} = 10 \\ \text{iterations} = 15\end{gathered}$$

From the third iteration onwards, the results indicate that by selecting objects 5, 6, and 7, we can achieve a total value of $9,050,000$, which is the best solution to our problem with a total weight of 9.5 kg.
However, if we want to analyze the behavior and convergence points of the data, it would be beneficial to use a number greater than 50 for the iterations. This would allow us to see the convergence points, representing the consolidation of a preferred path for all ants that perform the algorithm (the probabilistic factor is still present, but the likelihood of its occurrence is so low that it can be considered highly unlikely). Results for 15 iterations:

Objects: 0, 6, 4, 1; Profit: 7,050,000; Weight: 10
Objects: 1, 6, 4, 0; Profit: 7,050,000; Weight: 10
Objects: 6, 5, 4; Profit: 9,050,000; Weight: 10
Objects: 4, 5, 6; Profit: 9,050,000; Weight: 10
Objects: 5, 6, 4; Profit: 9,050,000; Weight: 10
Objects: 6, 5, 4; Profit: 9,050,000; Weight: 10
Objects: 6, 5, 4; Profit: 9,050,000; Weight: 10
Objects: 6, 5, 4; Profit: 9,050,000; Weight: 10
Objects: 4, 5, 6; Profit: 9,050,000; Weight: 10
Objects: 5, 6, 4; Profit: 9,050,000; Weight: 10
Objects: 6, 5, 4; Profit: 9,050,000; Weight: 10
Objects: 6, 5, 4; Profit: 9,050,000; Weight: 10
Objects: 6, 5, 4; Profit: 9,050,000; Weight: 10
Objects: 5, 6, 4; Profit: 9,050,000; Weight: 10
Objects: 4, 5, 6; Profit: 9,050,000; Weight: 10

Plots of the results:

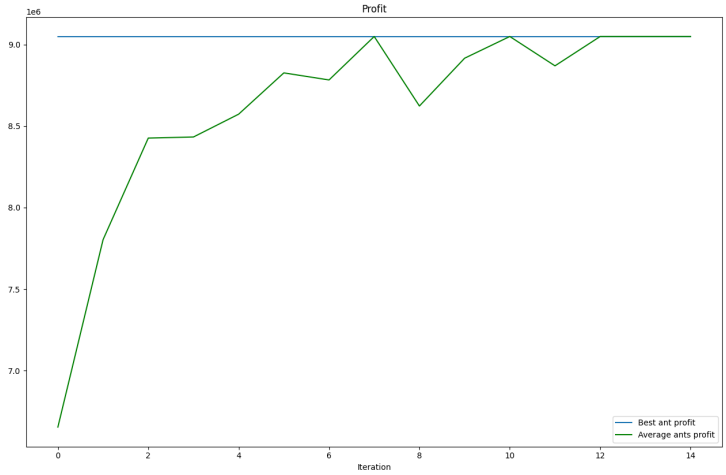

Figure 1: Average profit for iteration

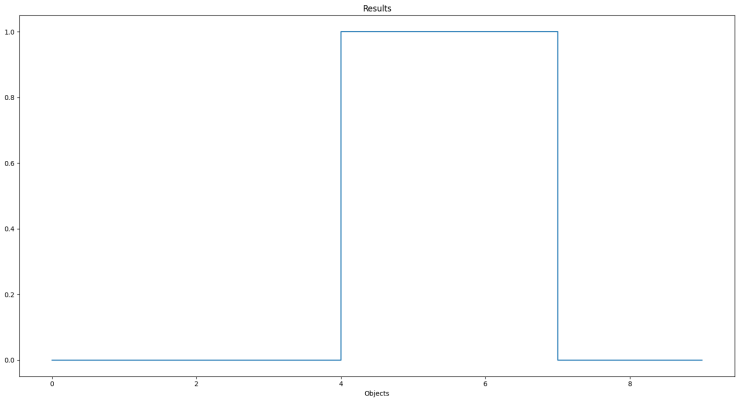

Figure 2: Selected items for the best solution

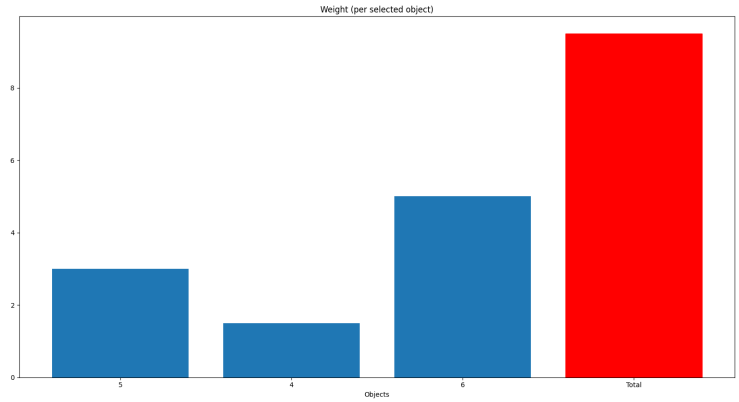

Figure 3: Selected items with weights and total weight

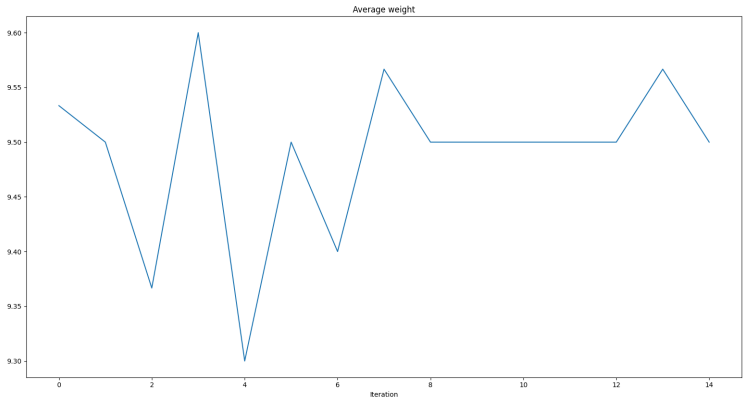

Figure 4: Average weight for iteration

Finally, I can conclude that the implementation of this code, expands the way we perceive problems, as it goes beyond what someone might commonly imagine for a solution. For the knapsack problem, the implementation of ACO is particularly effective as the problem possesses all the necessary characteristics to be solved by it. Personally, despite understanding ACO in the usual manner, attempting to comprehend how it could be adapted to a non-conventional problem proved to be a task that required time and effort. Nevertheless, I now have a broader knowledge of the subject.

4 Bibliography

- Schiff, K. (2013).Technical Transactions Automatic Control.

- https://github.com/astrastrastra/ACO-AS/tree/main