Genetic Algorithm for Traveling Thief Problem

Sang Dang Phuoc, Loc Ngo Cao, Hai Dang Quang

University of Information Technology, Vietnam National University, HCMC

Email: {21521377, 21521088, 21520027}@gm.uit.edu.vn

Abstract

Traveling Thief Problem (TTP) is a relatively new benchmark problem created to study problems that consist of interdependent subproblems. It is a combination of two well-known problems, the Knapsack Problem (KP) and the Travelling Salesman Problem (TSP). Euclidian 2D Traveling Salesman instances are combined with 0-1-Knapsack instances to reflect aspects of problems from the real world; for example, the total weight of the items in the knapsack influences the travel speed of a traveler. This introduced interdependence sets our benchmarks apart from capacitated vehicle routing problem instances, where this interdependence does not exist. In this thesis, we use Genetic Algorithm to solve this problem. The data we use to test our approach is from the TTP 2017 Competition - Optimisation of Problems with Multiple Interdependent Components.

Index Terms - Traveling Thief Problem, Travelling Salesman Problem, Knapsack Problem, Genetic Algorithm, TTP 2017 Competition.

1. Introduction

People are facing real-world problems with dependencies and interwovenness every day. Some researchers claim that there is a gap between theory and practice because either the problem or the benchmarks do not reflect real-world characteristics [2]. Therefore, to increase the complexity of benchmarks, problem instances are scaled up to come closer to real-world problems.

The Traveling Thief Problem (TTP) was introduced in 2013 by Bonyadi, Michalewicz & Barone (2013). It is a relatively new optimization problem as it is the combination of two different optimization problems. This problem is introduced by researchers when they are trying to compare different metaheuristics that take place from the perspective of NP-hard optimization problems. In the TTP, a person (thief) travels from a city and makes a round trip through the available cities. There are multiple items present in each city, which have some weight and value. The thief has a knapsack with a limited capacity and wants to collect the items available in different cities, which maximizes his profit without exceeding the knapsack capacity [3].

Genetic Algorithm (GA) is a search-based technique that has been used in optimization fields for practical problems. The idea behind the algorithm itself is based on the natural selection given by Charles Darwin's evolutionary synthesis in the 1860s. The basic concept for GA can be understood as the well-performed result will remain in the population and produce their better successor continuously, while the worse performance is removed. And the process continues until the inappropriate individuals eventually become extinct [4].

The rest of this thesis is organized as follows. We will present the background and definitions in section 2. It includes an introduction to optimization in general and defines single as well as multi-objective problems. Moreover, the two well-known problems, the TSP and KP, are explained.

These two problems are combined into the TTP. We also describe our approach to solving this problem in section 3. After that, section 4 will display our results with the data from TTP 2017 Competition. Followed by section 5, we will talk more about our limitations and ways to minimize them in our future work. Finally, we will have our conclusion for the whole thesis.

2. Background

2.1 Optimization

Optimization is done to make something as effective as possible. This can be useful in many situations and not least in our everyday life. In the following, definitions and principles of optimization are explained.

2.1.1 Single-Objective Optimization

Single-objective optimization problems are defined according to Equation. A function $f(\vec{x})$ is minimized or maximized. As parameter $\vec{x}$ is given, where the search domain is limited by the lower x_i^L and upper bounds x_i^U of each x_i. Furthermore, inequality constraints $g_j(\vec{x})$ and equality constraints $h_k(\vec{x})$ could be defined.

$$
\begin{aligned}
min/max \quad & f(\vec{x}) \\
s.t. \quad & g_j(\vec{x}) \geq 0 && j = 1, 2, ..., J; \\
& h_k(\vec{x}) = 0 && k = 1, 2, ..., K; \\
& x_i^L <= x_i <= x_i^U && i = 1, 2, ..., N;
\end{aligned}
$$

However, the limitation is that only one objective can be optimized. If several objectives exist, they have to be composed to one objective using a composing function. A naive approach is to define a weight for each objective and calculate the weighted sum.

2.1.2 Multi-Objective Optimization

The definition in Equation 2.1.2 is similar to Equation 2.1.1 But instead of only one function. Multiple functions $f_m(\vec{x})$ are considered. A single-objective is equal to a multi-objective problem when M = 1

$$
\begin{aligned}
min/max \quad & f_m(\vec{x}) && m = 1, 2, ..., M; \\
s.t. \quad & g_j(\vec{x}) \geq 0 && j = 1, 2, ..., J; \\
& h_k(\vec{x}) = 0 && k = 1, 2, ..., K; \\
& x_i^L <= x_i <= x_i^U && i = 1, 2, ..., N;
\end{aligned}
$$

2.2 Traveling Salesman Problem

In the TSP [5] a salesman has to visit n cities. The distances are given by a map represented as a distance matrix A = (d_{ij}) with i, j ϵ {0, ..., n}. The salesman has to visit each city once and the result is a permutation vector π = (π_1, π_2, ..., π_n), where π_i is the i-th city of the salesman. The distance between two cities divided by a constant velocity v results in the traveling time for the salesman denoted by f (π). The goal is to minimize the total traveling time of the tour:

$$
\begin{aligned}
min \quad & f(\pi) \\
s.t. \quad & \pi = (\pi_1, \pi_2, ..., \pi_n) \in P_n \\
f(\pi) \quad = \quad & \sum_{i=1}^{n-1} \frac{d_{\pi_i, \pi_{i+1}}}{v} + \frac{d_{\pi_n, \pi_1}}{v}
\end{aligned}
$$

There are $\frac{(n-1)!}{2}$ different tours to consider, if we assume that the salesman has to start from the first city and travels on a symmetric map where $d_{ij} = d_{ji}$.

2.3 Knapsack Problem

For the KP [6] a knapsack has to be filled with items without violating the maximum weight constraint. Each item j has a value $b_j \geq 0$ and a weight $w_j \geq 0$ where $j \in \{1, ..., m\}$. The binary decision vector $z = (z_1, ..., z_m)^2$ defines whether an item is picked. The aim is to maximize the profit g(z):

$$\begin{aligned} max \quad & g(z) \\ \text{s.t.} \quad & \sum_{j=1}^{m} z_j w_j \leq Q \\ & z = (z_1, .., z_m) \in \mathbb{B}^m \\ g(z) \quad = \quad & \sum_{j=1}^{m} z_j b_j \end{aligned}$$

The search space of this problem contains 2^n combinations.

2.4 Traveling Thief Problem

The TTP combines the above-defined subproblems and lets them interact together. The traveling thief can collect items from each city he is visiting. The items are stored in a knapsack carried by him. In more detail, each city π_i provides one or multiple items, which could be picked by the thief. There is an interaction between the subproblems: The velocity of the traveling thief depends on the current knapsack weight w, which is carried by him. It is calculated by considering all cities visited so far and summing up the weights of all picked items. The weight at city i given π and z is calculated by:

$$w(i, \pi, z) = \sum_{k=1}^{i} \sum_{j=1}^{m} a_j(\pi_k)\, w_j z_j$$

The function $a_j(\pi_k)$ is defined for each item j and returns 1 if the item could be stolen at city π_k and 0 otherwise. The current weight of the knapsack influences the velocity.

he speed of the thief is within a specified range [vmin, vmax], and it is always non-negative for a feasible solution. In the case of infeasibility (when the weight of the backpack exceeds Q), in order to provide the required travel time, we set the speed to vmin.

$$v(w) = \begin{cases} v_{max} - \frac{v_{max} - v_{min}}{Q} \cdot w & \text{if } w \leq Q \\ v_{min} & \text{otherwise} \end{cases}$$

The travel time of the thief will be calculated as follows:

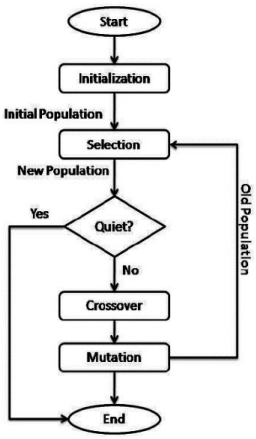
$$f(\pi, z) = \sum_{i=1}^{n-1} \frac{d_{\pi_i, \pi_{i+1}}}{v(w(i, \pi, z))} + \frac{d_{\pi_n, \pi_1}}{v(w(n, \pi, z))}$$

We observe that v is defined as a function (distinct from TSP), which takes the current weight w, dependent on i. Both v and w vary depending on z. Therefore, the solution to a subproblem of TSP depends on the solution to another subproblem, which is KP.

The objective of the problem is to minimize f(π, z) and maximize g(π, z):

$$G(\pi, z) = \begin{cases} min & f(\pi, z) \\ min & -g(\pi, z) \end{cases}$$

3. *Methods*



Since the Traveling Thief Problem is a NP-hard problem with sophisticated constraints, there is no obvious optimal solution to the problem itself. In this thesis, Genetic Algorithm is proposed as a promising solution.

The Genetic Algorithm (GA) is a powerful tool for solving optimization problems. The goal of TTP is to find the shortest route that visits all the cities and steals the maximum amount of loot, so it can be used to solve both KP and TSP. This is the flow chart of GA:

The first step is to initialize the population randomly. This means creating a set of possible solutions to the problem, each of which is represented as a chromosome. A chromosome is a sequence of genes, where each gene represents a city in the TSP. In the KP, each chromosome is a binary string, where each bit represents whether or not an item is included in the knapsack. The second step is to determine the fitness of each chromosome. The fitness of a chromosome is a measure of how good it is as a solution to the problem. In the case of the KP, the fitness of a chromosome is typically the total value of the items included in the knapsack, subject to the constraint that the total weight of the items does not exceed the knapsack capacity. In the case of the TSP, the fitness of a chromosome is typically the total distance of the route it represents. In the third step, we select parents for crossover and perform the crossover. The parents are the chromosomes that will be used to create new chromosomes in the next generation. The parents are typically selected based on their fitness, with the fittest chromosomes being more likely to be selected. Crossover is the process of combining two chromosomes to create a new chromosome. In both TSP and KP, the crossover is typically performed by swapping genes between two chromosomes. Next, we mutate some of the chromosomes. Mutation is the process of randomly changing a chromosome. Mutation can help to introduce new solutions into the population and prevent the algorithm from getting stuck in a local optimum. The final step is repeating steps 2-5 until a stopping criterion is met. The stopping criterion could be a fixed number of generations, a certain level of fitness, or a combination of both.

There are some hyperparameters for TSP and KP:

- For TSP: The maximum number of generations is 200 to make sure our algorithms converge. We also use a high crossover (0.99) probability that can lead to a faster convergence on a solution and a small mutation probability (0.01) to ensure only a tiny fraction of the population is mutated at each generation. The number of chromosomes is 120.

- For KP: We use a higher max generation (500) and mutation probability (0.5) to encourage the exploration of the search space. The population size is 15.

4. Results

4.1 Settings

We use the sample instances from the TTP 2017 Competition - Optimisation of Problems with Multiple Interdependent Components [7] to evaluate our methods. We choose 6 files to run our algorithms:

- 280_1.txt: include 280 cities, 1 item per city, lowest knapsack capacity, strongly correlated item weights/profits.

- 280_5.txt: include 280 cities, 5 items per city, medium knapsack capacity, uncorrelated but similar item weights/profits.

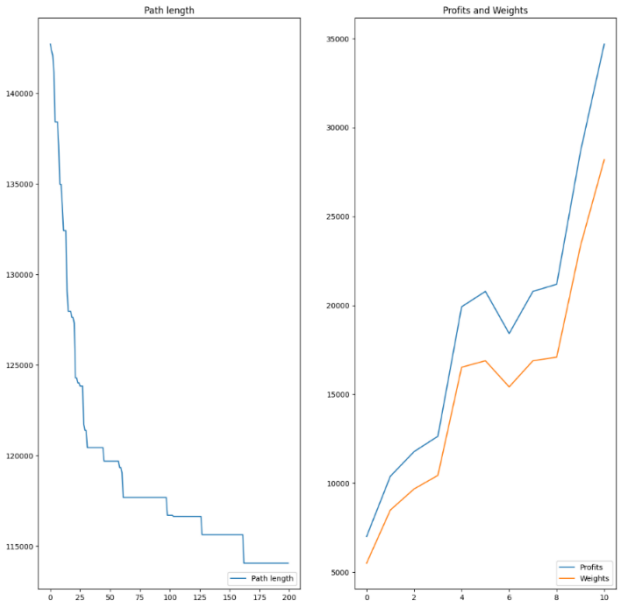
- 280_10.txt: include 280 cities, 10 items per city, highest knapsack capacity, uncorrelated item weights/profits.

- 4461_1.txt: include 4461 cities, 1 item per city, lowest knapsack capacity, strongly correlated item weights/profits.

- 4461_5.txt: include 4461 cities, 5 items per city, medium knapsack capacity, uncorrelated but similar item weights/profits.

- 4461_10.txt: include 4461 cities, 10 items per city, highest knapsack capacity, uncorrelated item weights/profits.

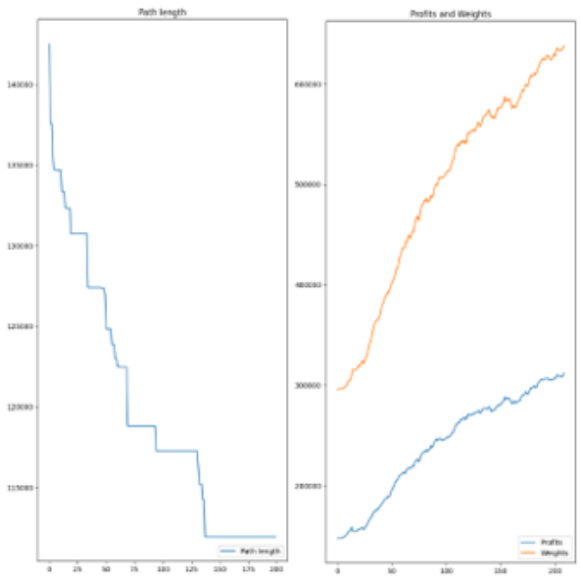
Here are the results:



*280_1.txt

TSP: shortest path: 114068.0

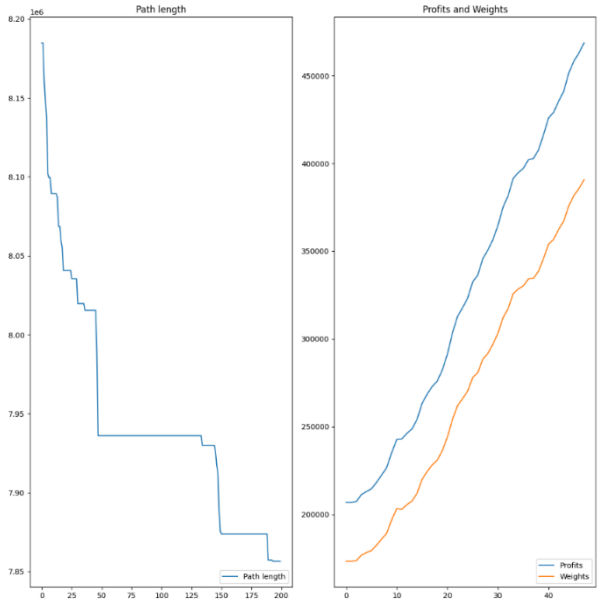
KP: maximum profit: 28641.0, weight: 23341.0



*280_5.txt

TSP: shortest path: 111936.0

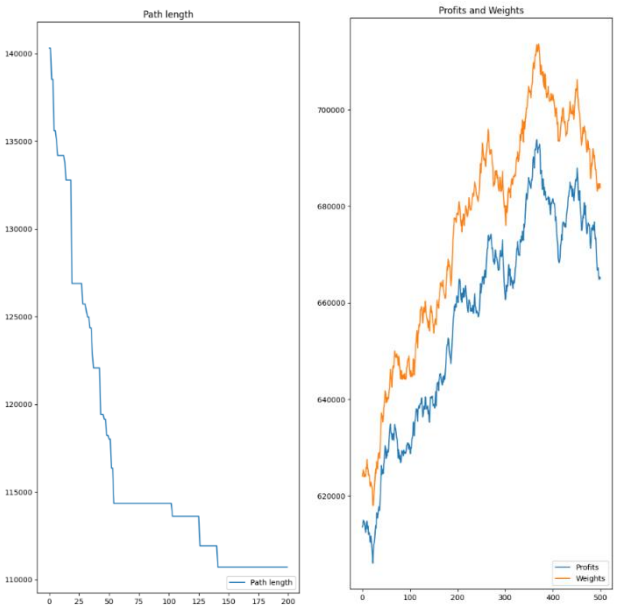
KP: maximum profit: 308608.0, weight: 634928.0



*280_10.txt

TSP: shortest path: 110703.0

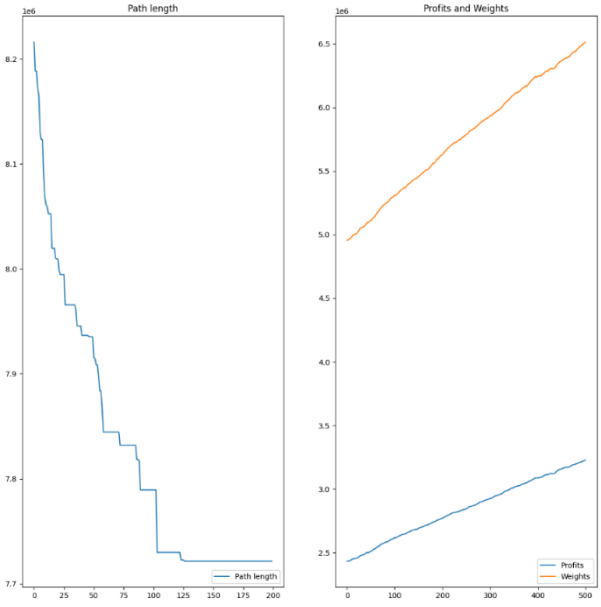
KP: maximum profit: 693760.0, weight: 713429.0



*4461_1.txt

TSP: shortest path: 7856582.0

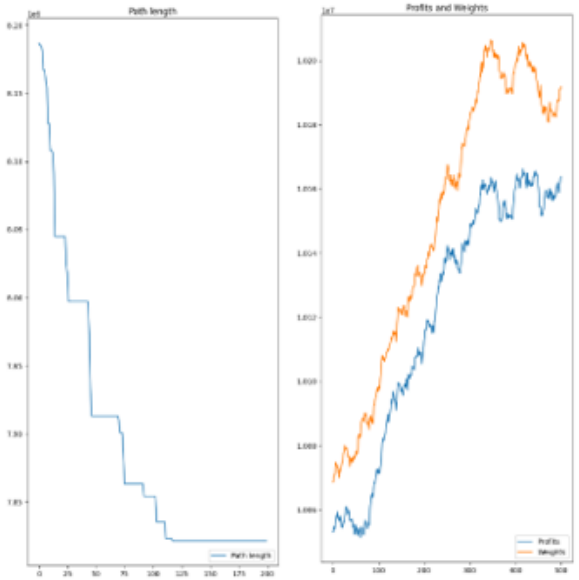
KP: maximum profit: 462973.0, weight: 385673.0



*4461_5.txt

TSP: shortest path: 7721430.0

KP: maximum profit: 3227710.0, weight: 6514221.0



*4461_10.txt

TSP: shortest path: 7821343.0

KP: maximum profit: 10166200.0, weight: 10205634.0

4.2 Result analysis

Our algorithm's performance is excellent with good results. However, the results are not surely optimal, especially when the weights and profits of items are uncorrelated.

It takes much more time to compute when the number of cities and the number of items are higher even using Google Collaborator. Therefore, we do not use higher sample instances in TTP 2017 Competition.

5. Future work

In this thesis, due to time pressure, we did not do a specific explanation of Genetic Algorithm and a test on a larger sample instance like 33810 cities. In addition, we did not find the optimal results in our test case to compare with our methods, so our results analysis is not clear.

In the future, we will tune our hyperparameters and improve our settings to get better results. Besides, we will do some experiments in larger test cases and research better methods for this problem.

6. Conclusion

In this work, we have a deeper understanding of the Genetic Algorithm and the combination of several NP-hard optimization problems that interact with each other. The Traveling Thief Problem is an example of a real-world optimization problem that combines two smaller subproblems: the Knapsack Problem and the Travelling Salesman Problem. We also introduce our method using Genetic Algorithm to solve these two subproblems and get good results in testing using sample instances from TTP 2017 Competition.

Although our results are not surely optimal solutions, with the background knowledge from this thesis, we can improve our algorithm and find out better ways to this problem.

References

[1] Rogier Hans Wuijts. *Investigation of the Traveling Thief Problem*. June 11, 2018. (cited on Page 2)

[2] Mohammad Reza Bonyadi, Zbigniew Michalewicz, and Luigi Barone. The traveling thief problem: The first step in the transition from theoretical problems to realistic problems. In *IEEE Congress on Evolutionary Computation*, pages 1037–1044. IEEE, 2013. (cited on Pages 1, 9, 10, and 17)

[3] Hamid Ali, corresponding author Muhammad Zaid Rafique, Muhammad Shahzad Sarfraz, Muhammad Sheraz Arshad Malik, Mohammed A. Alqahtani, and Jehad Saad Alqurni. A novel approach for solving the traveling thief problem using enhanced simulated annealing. In *PeerJ Computer Science*, March 16, 2021. (cited on Page 2)

[4] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT: Press, 1998.

[5] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton University Press, Princeton, NJ, USA, 2007. (cited on Page 6)

[6] Michail G. Lagoudakis. The 0-1 knapsack problem – an introductory survey, 1996. (cited on Page 7)

[7] TTP 2017 Competition - Optimisation of Problems with Multiple Interdependent Components. https://cs.adelaide.edu.au/~optlog/TTP2017Comp/ (accessed June, 2023)