THE TRAVELLING THIEF PROBLEM

ABSTRACT

The travelling thief problem is a hybrid of the knapsack problem and the travelling salesman problem. Some of the variables that influence the interdependence of these two sub-problems have been generated. There are two sets of parameters introduced, resulting in the production of two instances of **the travelling thief** problem.

INTRODUCTION

In the TTP, a person (thief) travels from one city to another, making a round trip via the cities that are available. Each city has an item with particular weight and value. The thief has a limited capacity knapsack and wishes to collect stuff available in various towns to maximize his profit without surpassing the knapsack's capacity. When collecting objects, a thief strives to select ones that are more valuable and lighter in weight to get the most money. However, note that by gathering the items, a thief's velocity will decrease as the empty knapsack capacity increases, affecting the tour's overall time. Furthermore, the thief must pay rent for his knapsack, which is proportional to the length of the excursion. The thief's overall profit is limited due to this issue.

PROBLEM ANALYSIS AND SUBDIVING INTO SUB PROBLEMS

The problem has been divided into two sub problems

TRAVELLING SALESMAN PROBLEM

In the TSP a salesman has to visit n cities. The distances are given by a map represented as a distance matrix A=(dij) with i,j∈{0,..,n}. The salesman has to visit each city once and the result is a permutation vector π=(π1,π2,...,πn), where πi 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

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

KNAPSACK PROBLEM

For the Knapsack Problem a knapsack must be filled with items without violating the maximum weight constraint. Each item j has a value bj≥0and a weight wj≥0 where j∈{1,..,m}. The binary decision vector z=(z1,..,zm) defines, if an item is picked or not. The search space of this problem contains *2^n* combinations and the goal is to maximize the profit g(z)

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

TRAVELLING THIEF PROBLEM

The TTP is a combinatorial optimization problem that consists of two interweaving problems, TSP and KNP. The traveling thief can collect one item from each city he is visiting. The items are stored in a knapsack carried by him. In more detail, each city provides one item, 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, which were visited so far, and summing up the weights of all picked items. The velocity v is always in a specific range v=[vmin,vmax] and could not be negative for a feasible solution. Whenever the knapsack is heavier than the

maximum weight Q, the capacity constraint is violated. However, to provide also the traveling time for infeasible solutions the velocity is set to vmin, if w>Q.

The calculation is based on TSP, but the velocity is defined by a function instead of a constant value. This function takes the current weight, which depends on the index i of the tour. The current weight, and therefore also the velocity, will change on the tour by considering the picked items. In order to calculate the total tour time, the velocity at each city needs to be known. For calculating the velocity at each city the current weight of the knapsack must be given. In fact, such problems are called interwoven systems as the solution of one subproblem highly depends on the solution of the other subproblems.

ALGORITHM

To solve this problem, we have taken the help of Evolutionary Algorithm for both sub problems i.e., The Travelling Salesman and the Knapsack problem. We have assumed that City1:'1' is the starting point of the thief. He always starts from City 1 and makes his tour to rest of the cities and is back to the starting point.
For our algorithm we defined few initial functions to generate distances between each city using random function, The weight of items in each city and the cost of each item is also generated using random function.
There are few other considerations like vmax=1, vmin=0.1 and C (max capacity of the knapsack)=20

For the First part of the problem (TSP): We have used the k-ary representation for a complete tour where a candidate solution (chromosome) is represented as a string of numbers, where each number represents the city he visited. For example, the chromosome: '1342' represents the tour of the thief, he starts at city1, then travels to city3 and then city4 and next city2. There are few constraints like no city can be visited twice and all cities must be visited. We have generated a random population of size 10, each representing a path taken by the thief.

For the Second part of the problem (Knapsack): We have used the k-ary representation for a complete tour where a candidate solution (chromosome) is represented as a string of bits where each bit represents whether he steals the item from the city, this is represented in a binary format: 0 for skipping the item in the city and 1 for stealing the item and putting in his knapsack. For example, the chromosome: '1010' represents the thief picks items from city he visits first and the city he visits third. He doesn't pickup anything from the city he visited second and last. We have generated a random population of size 10, each representing a picking order of the thief.

The ith chromosome for city tour and the ith chromosome for pickup mask is one solution and it is used to compute the time taken for the entire tour and the profit. The objective of the problem is to minimise the time taken and maximize the profit, by using a multi objective function: to maximize profit-time is used to define the fitness of the solution. The objective is to maximize the fitness.

The time taken to complete the tour from one city to another is computed using a heuristic function as defined in the previous section. It is the distance from one city to another divided by the velocity. Again, velocity is defined by:

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

Where w is the weight of the sack, Q is the maximum capacity of sack, vmax and vmin are the maximum and minimum velocity as defined in the starting of the algorithm.

We implemented Evolutionary Algorithm with the following parameters:

Parent Selection Mechanism: binary tournament is used for the solution (chromosome for city tour and pickup mask). We have also experimented with an alternative selector method: Choose best solutions as parents
Variation Operators:

1. Crossover: single locus crossover for the city tour chromosome and single point crossover for pickup mask chromosome.
2. Mutation operator: We also applied single point mutation for both the pickup mask chromosome and city tour chromosome,

Evaluation of Fitness: We then compute the time of tour and profit for the new solutions and use the multi-objective function to maximize profit-time ratio.
Survivor Selector mechanism (Replacement methods): Finally using a replacement method of replacing the weakest solutions with the new children solutions to update the population.
Termination Condition: Once 10,000 iterations of the EA is completed.

EXPERIMENTATION AND RESULTS

We have performed 10,000 iterations of the EA algorithm to find the best solution of city tour and pickup mask.

We have recorded the best solution at the end of one trial i.e., 10,000 iterations for both selector methods for number of cities= 7, 8 & 9. Below table has the records of the same:

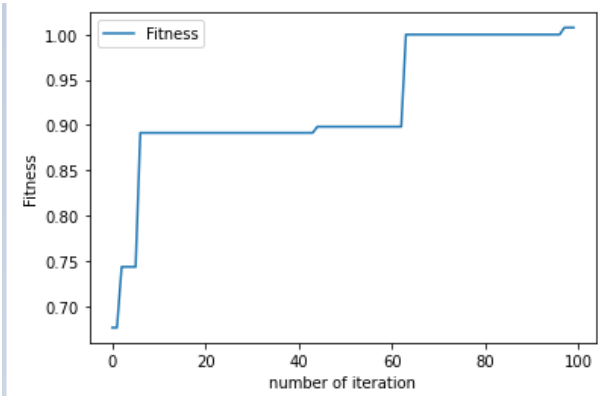
Best City Tour	Best pickup mask	Profit	Time taken	Profit Ratio=Profit/Time taken
1725364	000111	90	56	1.607142857
1543627	011111	131	130	1.007692308
18253674	0001011	80	70	1.142857143
17863452	0000111	92	63	1.46031746
173652894	11111111	255	160	1.59375
195786432	00001111	109	65	1.676923077

The above table shows the best solutions, i.e. optimal solution where Profit is maximum and time taken is minimum at the end of one trial or 10,000 iterations. It shows the best path taken by the thief and what should be his decision on stealing in his tour to obtain the maximum profit and minimum trip time. The path and pickup mask determine the velocity of the thief as he travels from city to city which in turn gives us the time. I have computed the profit to time ratio which shows the fitness of each solution, higher the ratio fitter is the solution.
Below results were recorded for fitness function Profit-Time:

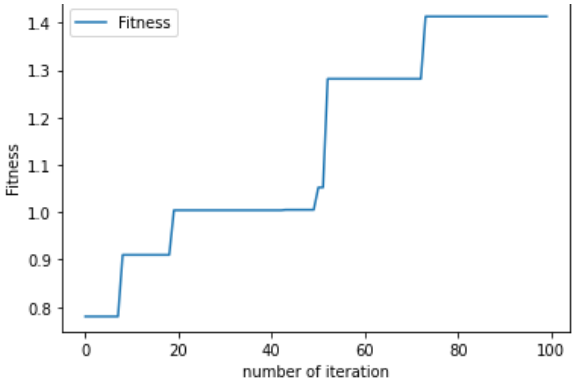
Parent Selection Type	Number of Cities	Best City Tour	Best pickup mask	Profit	Time taken	Net profit=Profit-Time taken
Best as parents	7	1436527	100011	89	82	7
Binary Tournament	7	1246375	011111	141	116	25
Best as parents	8	12457863	1000111	131	98	33
Binary Tournament	8	17682435	0000111	109	80	29
Best as parents	9	126493857	11111111	245	160	85
Binary Tournament	9	163498275	10111111	245	135	110

Below graphs represents the change of fitness for each of the algorithms for number of cities

Number of cities=7

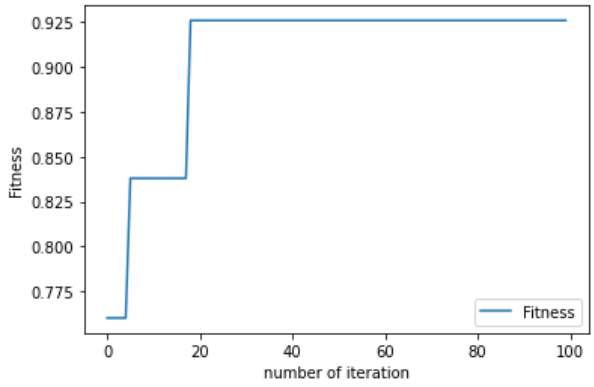


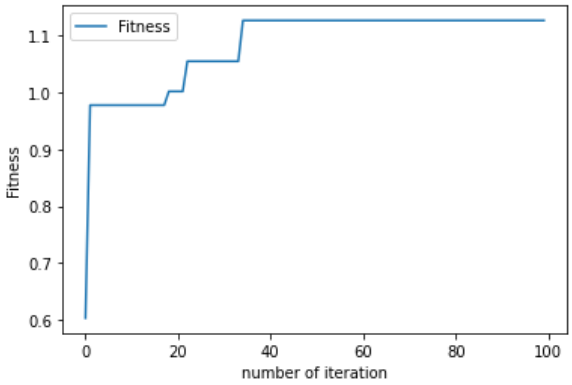
Binary Tournament selector



Best parents selector

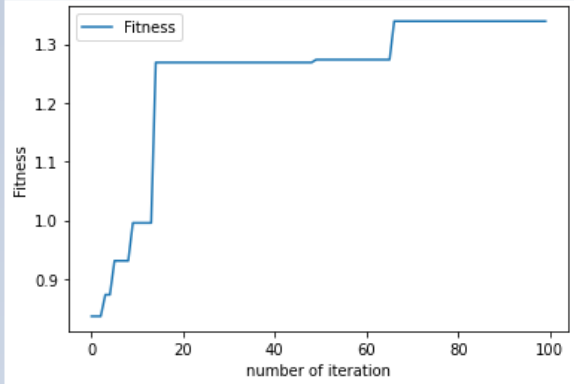
Number of cities=8

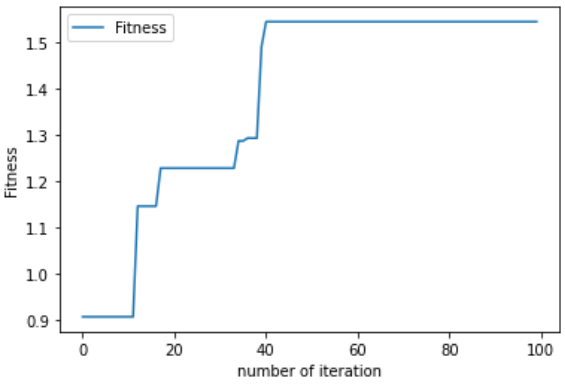




Binary Tournament selector **Best parents selector**

Number of cities=9





Binary Tournament selector **Best parents selector**

For both the algorithms we can see there is a rapid improvement in the fitness in the initial iterations and quickly converge to an optimal solution. As the number of cities increase the solutions' fitness increases rapidly at first and then converges.

RESEARCH UNDERTAKEN

We have conducted intensive research to solve the TTP using various scientific papers and citations published online and on several scientific journals for this problem. There were multiple approaches to solve the problem, the most popular ones were greedy algorithm, genetic algorithms, Evolutionary algorithm, simulated annealing, tabu search and ant colony optimization. There were several papers which referred to for finding the best algorithm for solving the are:

- Solving the Bi-Objective Traveling Thief Problem with Multi-Objective Evolutionary Algorithms, Authors: Julian Blank, Kalyanmoy Deb and Sanaz Mostaghim (WWW home page: http://www.is.ovgu.de/Team/Sanaz+Mostaghim.html).
- Hybrid Evolutionary Algorithm for Travelling Thief Problem by Neeraj Pathak, Rajeev Kumar
- A case study of algorithm selection for the travelling thief problem by Markus Wagner, Marius Lindauer, Mustafa Misir, Samadhi Nallaperuma, Frank Hutter
- Evolutionary Computation plus Dynamic Programming for the Bi-objective Travelling Thief Problem by Junhua Wu, Sergey Polyakovskiy, Markus Wagner, Frank Neumann

Ant colony optimization implements a meta heuristic approach to solve the TSP, but for the Knapsack problem ACO might not be easy to implement using ACO. After going through all the citations and papers we decided on using Evolutionary Algorithm as they provide competitive results with ACO and other nature inspired algorithms. In order to optimize the time for providing the solution and also minimize the complexity of the algorithm we chose the evolutionary approach to solve this problem.

CONCLUSION

This report has presented the solution of Travelling Thief problem using Evolutionary Algorithm with multiple parameters like population size of 10, with binary tournament selection method, choosing fittest solutions as parents, crossover recombination and mutation methods and worst solution replacement methods. It also shares the best solutions post each trial.