CA2 Group Project Report

Group M

December 2023

1 Introduction

This paper discusses the implementation of evolutionary algorithm (EA) for optimising the travelling thief problem (TTP)[2]. The Traveling Thief Problem (TTP) is an extension of the classical Traveling Salesman Problem (TSP) that incorporates elements of the Knapsack Problem (KP). The Traveling Thief Problem (TTP) poses a challenging combinatorial optimization task. In this study, we research into the application of four different optimisation algorithms- Ant Colony Optimization (ACO), Non-dominated Sorting Genetic Algorithm II (NSGA-II), Greedy algorithm, and Particle Swarm Optimization (PCO). During our research we leverage existing research papers to analyse and compare the algorithms' performances in solving the TTP. By reviewing a diverse set of research papers, we extract insights into the strengths and weaknesses of each algorithm with respect to solution quality, convergence speed, and robustness across various TTP instances. During our research, we identified NSGA-II as the algorithm exhibiting consistent superiority in terms of solution quality, convergence speed, and robustness across diverse TTP instances. Subsequently, we adopted NSGA-II to address the optimization challenge of the nine experiments. The results of these experiments provide valuable insights into NSGA-II's efficacy in solving the TTP across different scenarios and highlights practical implications in using NSGA-II for complex combinatorial optimisation challenges. By utilizing NSGA-II to successfully solve nine experiments, our study underscores its reliability and effectiveness in addressing the intricacies of the Traveling Thief Problem.

Over the past 50 years, researchers from diverse fields, including applied mathematics, operations research, metaheuristics, and artificial intelligence, have extensively studied a multitude of optimization problems. Most of these problems fall under the class of NP-hard, implying that finding optimal solutions for "large" instances becomes computationally intractable. This category encompasses renowned challenges like the Traveling Salesman Problem (TSP), the Knapsack Problem (KP), the Vehicle Routing Problem (VRP), the Multiprocessor Task Scheduling Problem and many others. These problems often represent real-world industrial environments and as such, solving them optimally holds significant value. The Traveling Thief Problem (TTP) emerges as a captivating extension of the classic Traveling Salesman Problem (TSP) by incorporating elements of the Knapsack Problem (KP). In the TTP, a cunning thief embarks on a journey through a network of cities, seeking to pilfer valuable items from each location. The thief's objective is to maximize the overall worth of stolen goods while navigating the delicate trade-off between maximizing the value of loot and minimizing the travel time. This optimization challenge presents a formidable combinatorial problem due to the inherent interdependence between the routing and packing decisions.

2 Possible Techniques

2.1 PSO

Particle Swarm Optimization (PSO) stands as a nature-inspired optimization algorithm, drawing inspiration from the collective behavior observed in bird flocks or fish schools [13]. In this algorithm, each particle within the swarm represents a potential solution to the optimization problem at hand. PSO's mimicking of nature's cooperative behavior makes it a powerful and widely used approach for addressing optimization challenges in diverse fields.

During research on the implementation of PSO for the Traveling Thief Problem, it was observed that the limitations, such as a high convergence rate leading to entrapment in local optima and the challenge of finding the global best [13], became particularly pronounced. Additionally, a notable increase in computational load with rising dimensionality was identified as a significant drawback[11]. Consequently, considering these limitations, the decision was made to forego the utilization of Particle Swarm Optimization for addressing the Traveling Thief Problem in our study.

After a comprehensive analysis of the research papers, it was observed that particle swarm optimization yielded promising results. However, as the dimensionality of the problem increased, reaching the global optimum became more challenging and computationally expensive.

2.2 ACO

Ant Colony Optimization (ACO)[6] is a metaheuristic that performs good quality and versatility on a variety of optimisation problems. Inspired by real life ants and their ability to search for a shortest path between a food source and their nest by exchanging information via pheromones, ACO is usually implemented by a finite sized colony of artificial ants building solutions in the iteration and each ant will leave the pheromone after constructing a solution which will affect

the choice of the next ants. After finite above iteration, it will give us a best result optimised by ants. The detail about ACO is described in [7].

For the TTP in this study, [4, 3, 15, 16] have successively applied different ACO-based algorithms to this problem. [4] is the first approach showing very poor results due to its reliance on only using backtracking search to cope with the constraints of TTP. The second one in [3] used ACO as a hyper-heuristic, still performing worse than other approaches. In [15] and [16], the former integrates ACO with Forward Checking and Conflicted-Directed Backjumping to get a better result than earlier approaches and the latter proposes a new MOEA/D-ACO algorithm (originated from [9]) based on finite pheromone weights obtaining a more uniform solution set and better convergence. However, the former still needs to improve heuristic values and exploration of the solution space and the latter needs further study on the datasets specification problem, the dividing line of the number of algorithm iterations, the number of averages, and the solution of multi-dimensional targets.

2.3 ISA

The Independent Subproblem Algorithm (ISA) is a method used to solve the Traveling Thief Problem (TTP). This approach involves dividing the problem into two subproblems: the Traveling Salesman Problem (TSP) and the Knapsack Problem (KP). The algorithm then aims to find the best solution for each subproblem separately and then merges the solutions to obtain the final result for the entire problem. Comparing ISA and NSGA-II for the Traveling Thief Problem reveals differences in their problem-solving approaches. ISA handles the problem by splitting the problem into two separate problems which fails to capture the intricate relationship between the distance traveling and optimising the knapsack problem. This isolation may limit overall solution quality due to the lack of coordination between optimizing the travel route and item selection. On the other hand, NSGA-II addresses both TSP and KP simultaneously, enabling a more integrated and comprehensive solution. Its ability to explore a wider solution space and handle conflicting objectives often leads to better-quality solutions compared to ISA's segregated approach. In[1] ISA solves each part of the problem separately, while NSGA-II looks at the whole problem together. In conclusion, while ISA simplifies complex problems, NSGA-II's holistic strategy often proves more effective for intricate optimization challenges like the Traveling Thief Problem.

2.4 Greedy Algorithm

A greedy algorithm operates by choosing a local optimum at each step with the expectation that a succession of such selections will ultimately achieve a globally optimal solution. The Greedy Algorithm selects the solution which seems best at the present moment, ignoring subsequent outcomes. A specific criterion or standard, ordinarily the local optimality of the problem, determines this selection at each stage.The algorithm's innate greediness typically affords simplicity and efficiency; however, it does not warrant a globally optimal solution to any problem. The algorithm aims to achieve excellence at a global level by making locally optimal decisions. The design of greedy algorithms necessitates the selection of locally optimal strategies with precision to ultimately arrive at the overall optimal solution through a sequential progression.[8] The following are the general steps of a greedy algorithm : 1. Problem modeling: abstracting a problem into a series of local decision-making steps. 2. Define optimization objectives: Clearly define the optimization objectives of the problem and determine the criteria for local optimality of the problem. 3. Choose the greedy strategy: at each step, choose the decision that currently seems optimal, which should be locally optimal. 4. Iteration: Repeat step 3 until the problem is solved.

Greedy algorithms are simple and easy to comprehend, and are often less expensive in terms of implementation and execution time. They sequentially identify locally optimal solutions by breaking the problem down and creating a solution based on the optimal choice at each step. Nonetheless, while greedy algorithms can find solutions that are locally optimal, they may become entrapped in these solutions, leading to an inability to optimize solutions globally. Once a decision is made, the greedy algorithm does not retrace its steps and may overlook superior solutions. NSGA-II is capable of addressing multi-objective optimisation problems, discovering a worldwide Pareto optimal solution by integrating multiple objective functions. NSGA-II produces Pareto frontiers that offer a diverse array of solutions instead of just one ideal resolution. [1] Consequently, NSGA-II is appropriate for intricate issues and can tackle both travel and backpacking problems.

3 Problem Breakdown

The task was broken down between the group as described below. The group document, code and experiments were broken down into components and divided equally amongst the members.

Usama Sajjad Ahmed: Introduction to the problem, PSO research, Designing experiments(code) and their plottings
Daji Liang: ACO, 3-OPT, NSGA research; design and code of experiments, hypervolume and analysis of results
Hugo Hewitt: NSGA code, NSGA description, KSP solver, taking minutes
King Lok Lam: Code Framework, TSP Code (3-opt Algorithm), Algorithm Explanation
Priyanka Naithani: ISA Algorithm research, Design and code Experiments , plotting the results of the experiment.
Yifeng Wang: Greedy Algorithm research, Designing experiments,Output experiment results Code

4 Developed Algorithm

4.1 NSGA-II

Non-dominated Sorting Genetic Algorithm II(NSGA-II)[5] was proposed by Deb *et al* in 2000. It was proposed as a fast elitists evolutionary algorithm for solving multi-objective problems. NSGA-II addresses the criticisms of the previous NSGA[14] algorithm, which include; high computational complexity, lack of elitism and lack of sharing parameter. NSGA-II solves the high computational cost of NSGA by using a fast non-dominated sorting approach that achieves $O(MN^2)$, this improves on NSGA's $O(MN^3)$ computational complexity.

NSGA-II is made up of 3 main modules, fast non-dominated sorting approach, density estimation and crowded comparison operator.

In the non-dominated sorting algorithm every solution from the population is checked with a partially filled population for domination. To start the first solution in the population is kept in P'. After the setup, each member of the population, p, is compared with all the members of the set P'. If p dominates any solution, q, in P' then solution q is removed from P'. Otherwise, if solution p is dominated by any member of P' then solution p is ignored. If solution p is not dominated by any member of P' then it is entered into P'. When all members of the population have been checked P' ends up being the non-dominated set of solutions.

The density estimation is calculated from particular point in the population, from this point take the average distance of the two points either side of this point, along both objectives (time and profit). This is then used to create the largest cuboid around the point without including any other points. This cuboid is the crowding distance of the point.

The crowded comparison operator guides the selection process throughout the algorithm in order to create a uniformly spread-out Pareto-optimal front. They are compared as follows; 1. Non-domination rank (i_rank), 2. Local crowding distance ($i_{distance}$).

$i <_n j$ if ($i_{rank} < j_{rank}$) or (($i_{rank} = j_{rank}$)and($i_{distance} > j_{distance}$))

NSGA-II is the best implementation for our system as it is simple enough to be able to understand and implement within the time constraints of the deadline and complex enough to be able to effectively solve the travelling thief problem effectively. NSGA-II offers a balance between complexity and performance. NSGA-II has also been cited thousands of times and can be considered as one of the best nature inspired algorithm for multi-objective problems.

4.2 3-opt in TSP

Inspired by general-purpose iterative search-heuristic method[12] for the single-objective TTP, we finally decided to apply 3-opt to the part of TSP.

λ optimality:

λ optimality, a local search algorithm, is to classify tours into descending classes with stronger optimality conditions and a tour which is λ optimal (simply λ-opt) is not be able to be replaced any λ of its links by any other set of λ links given smaller cost. 3-opt examines different combinations of three edges in the path of initial solution and tries to obtain a shorter path by swapping their order.

Pros: [10] shows us repeated runs lead to a high probability of finding the optimal solution and the proportion of a 3-opt tour is evidently higher than that of 2-opt tours. Besides, the computation time of 3-opt is also considerably smaller than that of 4-opt with the probability of finding optimal solution which is not significantly reduced.

Cons: The average time required per locally optimal solution is under $30n^3$ microseconds where n is the number of cities involved[10], so we have to shorten the running time for larger number of cities in this dataset.

Application in the problem solver:

In a standard 3-opt algorithm, it takes $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ to go through all possible combinations in order to have a fully optimised solution. For instance, in the 280, 4461, 33810 nodes data sets. They have 3,619,560, 14,786,088,590 and 6,440,887,676,720 3-opt combinations respectively. It is impossible to run through all combinations in the 2nd and 3rd data sets, the running time will be astronomical. There will be not enough memory to store all possible combinations. Moreover, a distance matrix in the 3rd data set will take roughly 20GB of memory which is not memory efficient. We decided to take some trade off. Only $10000 + \pi$ 3-opt operations will be done in our algorithm. The running time for algorithm is slightly slower to save memory, due to real time calculation of the path distance.

5 Results

The experiment employed the fnl4461-n4460 dataset with the following parameter settings:

population size nsg: 20
evaluations tsp: 1000
run local tsp: True
tournament size ksp: 5
num generations ksp: 200
fill rate ksp: 1

Each member compared the crossover and mutation methods they used and gave the best combination. The most effective crossover and mutation methods, as determined by each individual's experiment, were then compared to the original methods and the results are shown below:

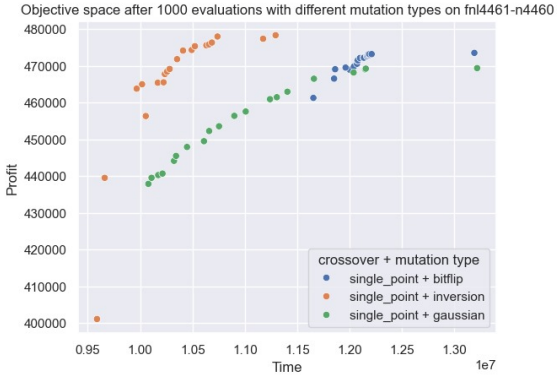

Figure 1

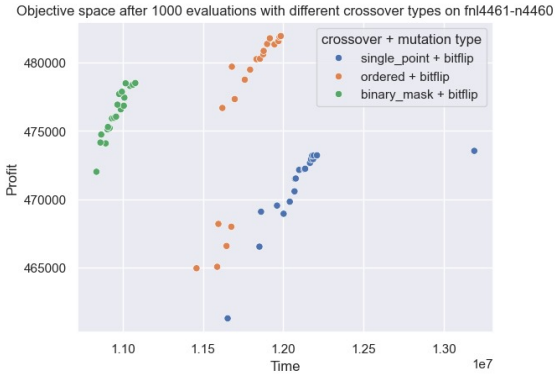

Figure 2

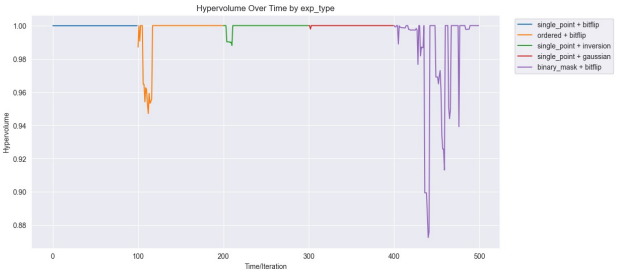

Figure 3

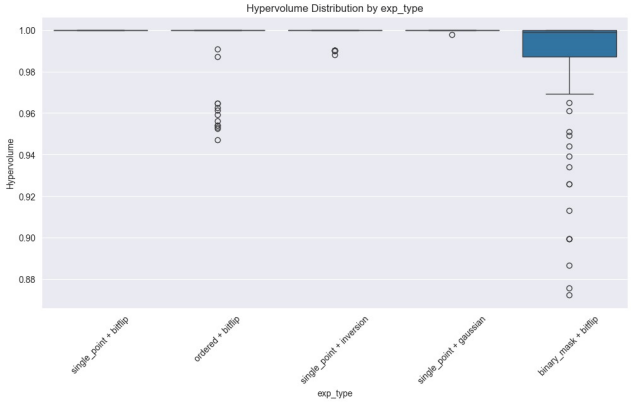

Figure 4

Figure 2 shows that Binary-mask crossover achieves a higher profit value in a shorter time, while Ordered crossover achieves the highest profit value in a slightly longer time. Figure 1 shows that Inversion mutation achieves the highest value in the shortest time. The values of Hyper Volume in Figures 3 and 4 indicate that the stability of Inversion mutation is the best.

References

[1] Blank, J., Deb, K., and Mostaghim, S. Solving the bi-objective traveling thief problem with multi-objective evolutionary algorithms. In *Evolutionary Multi-Criterion Optimization: 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings 9* (2017), Springer, pp. 46–60.

[2] Bonyadi, M. R., Michalewicz, Z., and Barone, L. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *2013 IEEE Congress on Evolutionary Computation* (2013), pp. 1037–1044.

[3] Chen, P.-C., Kendall, G., and Berghe, G. V. An ant based hyper-heuristic for the travelling tournament problem. In *2007 IEEE Symposium on Computational Intelligence in Scheduling* (2007), IEEE, pp. 19–26.

[4] Crauwels, H., and Van Oudheusden, D. Ant colony optimization and local improvement. In *Workshop of Real-Life Applications of Metaheuristics, Antwerp, Belgium* (2003), Citeseer.

[5] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *Parallel Problem Solving from Nature PPSN VI* (Berlin, Heidelberg, 2000), M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds., Springer Berlin Heidelberg, pp. 849–858.

[6] Dorigo, M. Ant colony optimization. *Scholarpedia 2*, 3 (2007), 1461.

[7] Dorigo, M., Di Caro, G., and Gambardella, L. M. Ant algorithms for discrete optimization. *Artificial life 5*, 2 (1999), 137–172.

[8] Gupta, B. C., and Prakash, V. P. Greedy heuristics for the travelling thief problem. In *2015 39th National Systems Conference (NSC)* (2015), pp. 1–5.

[9] Ke, L., Zhang, Q., and Battiti, R. Moea/d-aco: A multiobjective evolutionary algorithm using decomposition and antcolony. *IEEE transactions on cybernetics 43*, 6 (2013), 1845–1859.

[10] Lin, S. Computer solutions of the traveling salesman problem. *Bell System Technical Journal 44*, 10 (1965), 2245–2269.

[11] Oldewage, E. T., Engelbrecht, A. P., and Cleghorn, C. W. The merits of velocity clamping particle swarm optimisation in high dimensional spaces. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (2017), pp. 1–8.

[12] Polyakovskiy, S., Bonyadi, M. R., Wagner, M., Michalewicz, Z., and Neumann, F. A comprehensive benchmark set and heuristics for the traveling thief problem. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation* (2014), pp. 477–484.

[13] Song, M.-P., and Gu, G.-C. Research on particle swarm optimization: a review. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)* (2004), vol. 4, pp. 2236–2241 vol.4.

[14] Srinivas, N., and Deb, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput. 2*, 3 (sep 1994), 221–248.

[15] Uthus, D. C., Riddle, P. J., and Guesgen, H. W. An ant colony optimization approach to the traveling tournament problem. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (2009), pp. 81–88.

[16] Yang, L., Jia, X., Xu, R., and Cao, J. An moea/d-aco algorithm with finite pheromone weights for bi-objective ttp. In *Data Mining and Big Data: 6th International Conference, DMBD 2021, Guangzhou, China, October 20–22, 2021, Proceedings, Part I 6* (2021), Springer, pp. 468–482.