Nature Inspired Computation Group Project

Group N

1 Introduction

This group report focuses upon optimising the multi-objective Travelling Thief Problem (TTP) through the use of nature-inspired algorithms. The TTP is an intricate optimization problem that combines aspects of the Travelling Salesman Problem (TSP) and Knapsack Problem. The interdependence between these two objectives is responsible for its complexity, and as a result requires a solution that optimises the two. This report delves into the research undertaken, focusing on algorithm design, fitness functions, distance calculations, and the integration of both Ant Colony Optimization (ACO) and Local Search for enhanced solutions.

In order to conduct this project, we split the report into smaller relevant tasks: algorithm research, initial programming, algorithmic programming, testing and finally the compiling of results.

2 The Problem

Due to the bi-objective nature of the TTP, a solution that balances the two objectives is needed. Such a solution would have the attributes of a high profit (from the KP) whilst maintaining a low time (for the TSP). The interaction between the two problems occurs in the weight of items, w, carried by the thief, which affects the velocity, v, in which they travel. The velocity is calculated as such.

$$v = v_{max} - (v_{max} - v_{min})\frac{\tilde{w}}{Q} \tag{1}$$

When the knapsack is empty ($\tilde{w} = 0$) the speed is maximised (v_{max}), and when the knapsack is full ($\tilde{w} = Q$) the speed is minimised (v_{min}) [6]. Thus, if a thief is too greedy and only focuses upon profit, they will be too slow, but if the thief only focuses upon time, they will obtain no profit. We later present a novel solution to overcome this multi-objective challenge, .

In order to assess the proficiency of our algorithm, it is essential to define our fitness function. Based on the prior information, we calculate the time taken for a given route, based upon the distance as well as the weight of items selected. We also calculate the profits of the items stolen. As such, our final results can be represented in a two dimensional plot, whereby we aim to minimise time and maximise profit.

3 Separation of the Problems

Our approach in completing this project was to break it down into three key phases, in which the work was broken down into different parts for the team. We split into groups of two for most tasks that were executed.

The first was research and initialization. It was important to first gain an understanding of the problem, its requirements, and how others had addressed it. We also began writing some of the core functions, including loading the data and identifying the non dominating solutions. The second was the implementation of algorithms and testing. Here different subteams implemented different algorithms and tested their results against standard data sets, in order to aid our final selection. The final stage was the consolidation of the solutions and deliverable production. This involved putting together the different aspects of the project to create a complete report. We assigned the preparation of the different files we needed to submit to different team members, including results gathering, report writing and miscellaneous tasks such as organising the code.

4 Algorithmic Research

We collectively conducted a thorough research process in order to select the most appropriate algorithms for the problem. To do so, we adopted a comprehensive method which broke each potential algorithm into their fundamental principles, advantages, and drawbacks when applied to the TTP. We also considered the implementation difficulty of each algorithm, and its applicability in relation to the TTP. We researched and considered a wide variety of algorithms

One early method we investigated was Particle Swarm Optimisation (PSO). This algorithm mimics the behaviour of birds, whereby a flock moves together to benefit from the experience of the rest of the members [10]. Nevertheless, a classic PSO algorithm is not suited for a permutation based problem (our TSP and KP), and instead useful for continuous search spaces.

The next set of algorithms we evaluated for addressing the TSP were ruled out due to their unique limitations. Simulated Annealing (SA) relies heavily on parameter tuning and may face challenges in rapidly changing environments due to its inherent nature of stochastic search [3]. Memetic Algorithms, effective as they are, might incur higher computational costs,

especially in resource-intensive fitness evaluations [1]. Multi-Objective Evolutionary Algorithms (MOEAs) face difficulties in balancing conflicting objectives and may require careful consideration of Pareto front approximation techniques [11], and Hybrid Algorithms introduce complexity in parameter tuning with no guaranteed superior performance [8]. Variable Neighborhood Search (VNS) effectiveness is contingent on the choice of neighbourhood structures, impacting its adaptability [3]. Biogeography-Based Optimization (BBO) may lead to suboptimal solutions without careful parameter tuning [7], and the Firefly Algorithm's (FA) efficacy diminishes in complex optimization problems such as the TTP [4].

Upon our final review, we shortlisted our final algorithm selection to Ant Colony Optimisation (ACO), Genetic Algorithms (GA) and Local Search.

We implement the ACO to solve the TSP aspect of our problem. Our preference for this algorithm arises from its robustness in balancing exploration and exploitation, adaptability to dynamic problem instances, and insights derived from ant foraging behaviour [12].Another key advantage of the ACO was the ease to utilise the solution information from the solved KP. More specifically, we were easily able to adapt the heuristic within the ACO that calculates the probabilities for an ant taking a given path to use time, which is dependent on the weight of the items selecting when solving the KP. Finally, upon testing the algorithm for the Brazil58 data set, we found the results to improve upon those calculated by a GA, and in a substantially faster time.

For the Knapsack problem, we experimented with both Local Search and GA. The former is advantageous for solving the Knapsack problem in the Travelling Thief Problem due to its flexibility in exploring large solution spaces efficiently. Because of its flexibility, the algorithm can be adapted to utilise the advantage of particular features of the problem, which makes it a useful heuristic for combinatorial optimization problems. Its method of incremental improvement helps to maximise overall profit by gradually improving solutions [5]. Furthermore, it holds rapid computational speed, which is incredibly useful for the larger data sets.

Genetic algorithms are also useful for combinatoral optimisation problems. We note the key benefit of a GA for problems such as the KP is its ability to diversify its solution set using mutations, whilst maintaining the best elements of the previous generation [9]. We note results from a study which included a large number of variables successfully managed to find acceptable solutions in a short time [9], which is incredibly desirable for our larger data sets. We note a key advantage of the GA is its variability; we can adapt the mutation operators, including different swap operators, replacement functions and selection processes. While this is beneficial for improving results, to completely optimise a GA's results is timely, which is challenging given the tasks deadlines.

5 Experiments

We initially explored local search with various heuristics, including the profit-to-weight ratio and a focus on the knapsack problem. However, after rigorous testing, we ultimately opted for a genetic algorithm due to its superior performance. The chosen genetic algorithm incorporated the profit-to-weight ratio heuristic, guiding initial solution selection. Additionally, discretion was employed to exclude items with specific weight factors, ensuring a balanced item selection. While both heuristics enhanced algorithm performance, varying data set sizes required adjustments to the discretion value for optimal outcomes.

6 Final Algorithm

Our final algorithm splits the problem into the two sub problems: the KP and the TTP. The algorithm first conducts a local search to solve the KP, and then uses that information for solving the TSP. We use this method for a few reasons. This made the allocation of work much easier, allowing the core task to be split into the two problems for different team members to work on. In addition there is evidence to support the success of this approach in reducing the complexity and speeding up the search process, without sacrificing too much accuracy [6].

Once the genetic algorithm identifies a solution, we use an adapted ACO in order to use the information to dictate the route of the thief. This occurs in the edge selection formula, whereby the probability of a given ant taking a path is determined by the pheromones deposited as well as the time and profit of a given city to city connection. Thus, the distance of a given journey, as well as the items stolen at the next city influence the probability matrix. Finally, we include a local search in order to find to increase the number of available solutions.

7 Results

In this section, we compare the results of our algorithm across the nine different instances included in the competition. As previously discussed, the size of these data sets differ, and as such we utilise different parameters to assist with computational speeds. We acknowledge that this will impact the results we obtain.

We include the non-dominated set of solutions for each of the nine instances in Figure (1). These are the set of solutions that are non-dominated to each other yet are improvements to the rest of the points. Upon plotting this set of solutions against negative profit and time, we find that our algorithm has successfully minimised the two objectives – depicted by the convex Pareto frontier. This suggests the strength of our

Instance	# of Ants	# of Iterations	
		ACO	Local Search
a280	20	100	2000
fln4461	10	20	1000
pla33810	2	5	500

Table 1: Parameters for each algorithm by Instance

algorithm in effectively balancing the multi-objective nature of the TTP.

Across most of the instances, the results curve is incredibly similar. Nonetheless there are certain outliers. The algorithm seems to perform best for the smallest instance 'a280-n279', with the non-dominated solutions achieving a shorter time frame for the same level of profit when compared to the others. This plot also contains a few outlier solutions at the lower inverse profit levels.

We also note the consistency within the columns, which contain different item counts for the same city count. As such, we expect that if the algorithm is correctly balancing both time and profit, that the time taken should be consistent despite the increase in available items. Our results display such consistency, and whilst inverted profit levels do change, time for the same city counts remain within a similar region.

Despite the apparent strength of our algorithm, it is important to make some comparisons to those found in the competition. Across all instances we are outperformed by the teams in the competition, noted by the enhanced convexity of their Pareto fronts. Unexplored improvements to our algorithm, as well as a greater number of iterations may help improve our results.

8 Conclusion

Using nature inspired algorithms, we were successfully able to obtain strong results across all nine of the instances. The non-dominated sets of results suggests the strength of our algorithm in effectively balancing both aspects the multi objective TTP problem, by successfully minimising time whilst simultaneously maximising the profit.

As a team, we successfully conducted a comprehensive research project encompassing its various stages, from the initial research and investigation of algorithms, the implementation of algorithms and testing and finally the collation and analysis of the results. Throughout this collaborative process, we emphasized effective task delegation, ensuring all members contributed to the timely completion of the project to a high standard.

Whilst we carried out an exhaustive research phase, we acknowledge the room for further experimentation within our investigation. Namely, the opportunities to fine tune certain ACO parameters, as well as testing a wider range of algorithms. One example of a high performing algorithm is a Memetic Algorithm, which includes a local search heuristic to improve the solutions generated by a GA. Evidence suggests that this method is very effective in generating high performing solutions to the TTP, especially for small and medium sized instances [2].

References

[1] Marco Baioletti, Alfredo Milani, and Valentino Santucci. Algebraic particle swarm optimization for the permutations search space. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1587–1594. IEEE, 2017.

[2] Mohamed El Yafrani and Belaïd Ahiod. Population-based vs. single-solution heuristics for the travelling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 317–324, 2016.

[3] Rani Kumari and Kamal Srivastava. Variable neighbourhood search for bi-objective travelling thief problem. In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 47–51. IEEE, 2020.

[4] Jianxun Liu, Jinfei Shi, Fei Hao, Min Dai, and Zhisheng Zhang. A new firefly algorithm with enhanced attractiveness. In *2021 5th International Conference on Automation, Control and Robots (ICACR)*, pages 74–77. IEEE, 2021.

[5] Alenrex Maity and Swagatam Das. Efficient hybrid local search heuristics for solving the travelling thief problem. *Applied Soft Computing*, 93:106284, 2020.

[6] Yi Mei, Xiaodong Li, and Xin Yao. Improving efficiency of heuristics for the large scale traveling thief problem. In *Asia-Pacific Conference on Simulated Evolution and Learning*, pages 631–643. Springer, 2014.

[7] Hongwei Mo and Lifang Xu. Biogeography based optimization for traveling salesman problem. In *2010 Sixth International Conference on Natural Computation*, volume 6, pages 3143–3147. IEEE, 2010.

[8] Quang Huy Nguyen, Yew-Soon Ong, and Natalio Krasnogor. A study on the design issues of memetic algorithm. In *2007 IEEE Congress on Evolutionary Computation*, pages 2390–2397. IEEE, 2007.

[9] Richard Spillman. Solving large knapsack problems with a genetic algorithm. In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, volume 1, pages 632–637. IEEE, 1995.

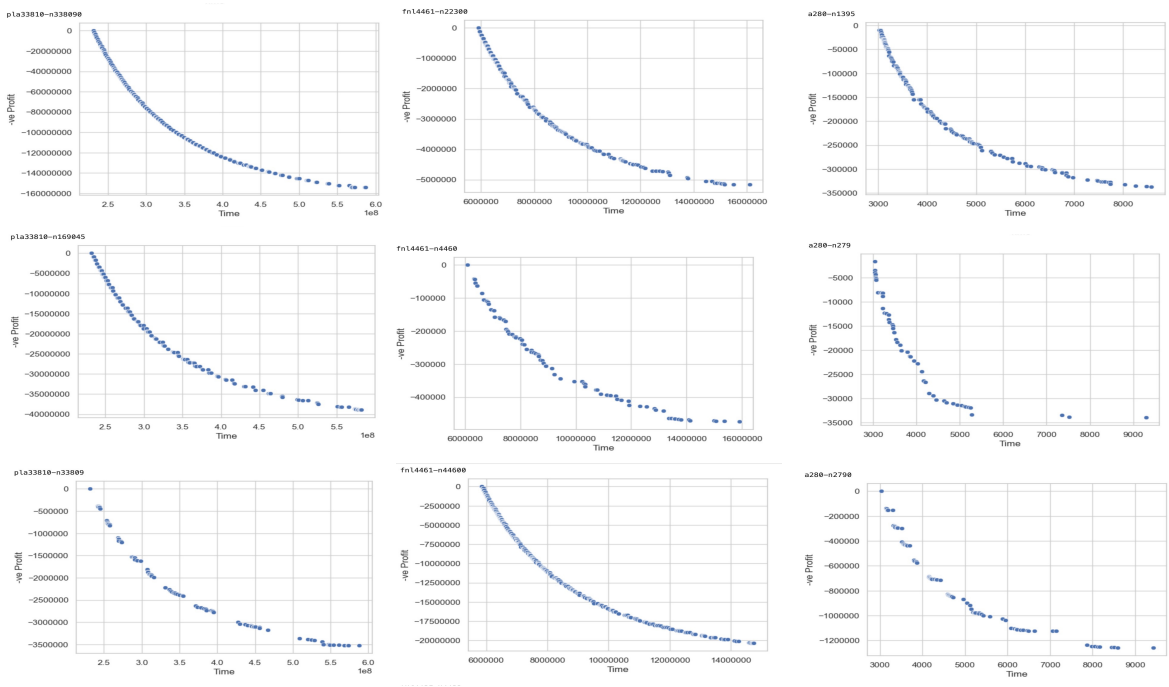

Figure 1: Inverse Profit by Time (seconds) for the Gecco2019 Competition Instances

[10] Adrian Tam. A gentle introduction to particle swarm optimization. *Machine Learning Mastery*, 2021.

[11] Vimal L Vachhani, Vipul K Dabhi, and Harshadkumar B Prajapati. Survey of multi objective evolutionary algorithms. In *2015 international conference on circuits, power and computing technologies [ICCPCT-2015]*, pages 1–9. IEEE, 2015.

[12] Wiem Zouari, Ines Alaya, and Moncef Tagina. A hybrid ant colony algorithm with a local search for the strongly correlated knapsack problem. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 527–533. IEEE, 2017.