Contents

Chapter 1

Introduction

(TODO) Set the scenes. Explain why you are doing this work and why the problem being solved is difficult. Most importantly you should clearly explain what the aims and objectives of your work are.

(TODO) Structure of the thesis. Academic publications produced (if any), including any achievements/highlights

Chapter 2

Literature Review

2.1 What is Optimization?

2.1.1 Definition and Concept

Discuss the general concept of optimization, emphasizing its role in operations research and decision-making, and explaining how it aims to find the most favorable solution from available alternatives.

2.1.2 Exact vs. Heuristic Methods

Exact Methods: Describe methods that provide guaranteed optimal solutions, such as linear programming and integer programming. **Heuristic Methods:** Introduce heuristic approaches, like genetic algorithms and simulated annealing, used in complex problems where exact solutions are computationally impractical. **Comparison:** Analyze the trade-offs between exactness and computational efficiency.

2.1.3 Day-to-Day Optimization Examples

Illustrate how optimization is applied in everyday scenarios such as route planning, airline scheduling, and personal time management.

2.2 Building Blocks for Monte Carlo Tree Search

2.2.1 Markov Chains

Introduce the theory of Markov Chains and their application in modeling stochastic processes and optimization.

2.2.2 Monte Carlo Methods

Discuss the basics of Monte Carlo methods, focusing on their use in solving complex computational problems through random sampling.

2.3 Monte Carlo Tree Search

Detailed discussion on Monte Carlo Tree Search (MCTS), explaining how it extends Monte Carlo methods to decision-making problems, particularly highlighting its relevance in AI and game theory. Do different part to highlight the differents plociies, the different parameters - how it has been used especially in game theory and so on

2.4 Monte Carlo Tree Search in Routing Optimization

2.4.1 Overview of Routing Challenges

Present an overview of different routing problems, including the Traveling Salesman Problem and the Vehicle Routing Problem, focusing on their complexities and the optimization strategies employed.

2.4.2 Application to Kiwi.com Problem

Discuss the specific application of MCTS to the Kiwi.com challenge, including how the unique properties of the problem are addressed by this method.

2.5 Review of Previous Approaches to the Kiwi.com Problem

2.5.1 Survey of Approaches

Discuss main approaches like Local Search Algorithms, Reinforcement Learning, and Genetic Algorithms, mentioning key studies that have tackled similar challenges. **Local Search Algorithms:** Explore the principles and applications of Local Search, highlighting specific papers and their results. **Reinforcement Learning:** Review the application of Reinforcement Learning in transportation optimization, citing relevant studies. **Genetic Algorithms:** Describe how Genetic Algorithms have been used to solve complex routing problems and their effectiveness in various scenarios.

2.5.2 Comparative Analysis

Provide a comparative analysis of these methods with Monte Carlo Tree Search, evaluating performance under different conditions based on the nature of the problem.

2.6 Research Motivation and Literature Gaps

2.6.1 Motivation for This Study

Discuss the motivations for using MCTS in solving the Kiwi.com problem, considering the problem's complexity and scale.

2.6.2 Identification of Gaps

Highlight areas not fully explored by previous studies, such as scalability or real-time application constraints, emphasizing the novel aspects of your research.

2.7 Critical Analysis of Existing Research

Analyze the strengths and weaknesses of the studies reviewed, providing a critical perspective in relation to your research. Discuss how convincing the authors' arguments are and where your research might offer improved or alternative solutions.

(TODO) Further refine and expand sections based on feedback and ongoing research.

Chapter 3

Problem Description

(TODO) Present and explain your problem AHA in detail Assumptions should be stated clearly

3.1 Overview

Kiwi's traveler wants to travel in n different areas in n days, let's denote *Areas* the set of areas the traveler wants to visit: $Areas = \{\text{Area}_1, \text{Area}_2, \ldots, \text{Area}_n\}$.

For every existing areas, there is at least one airport that belongs to this area:

$$\forall i \in \text{Areas}, \exists\, j \in \text{Airports}\,\text{such that}\, j \in i \tag{3.1}$$

Without considering the order, we can say that the traveler is in Area_1 at day_1 at Airport_1.

The traveler has to visit one and only one area per day, no matter what airport he visits but he leaves from the starting airport and has to come back to the starting area (not necessary the startinga airport). There are flight connections between different airports, with different prices depending on the day of the travel: we can write c^d_{ij} the cost to travel from $city_i$ to $city_j$ at day d. We do not necessarly have $c^d_{ij} = c^d_{ji}$ neither $c^{d_1}_{ij} = c^{d_2}_{ij}$ if $d_1 \neq d_2$. The problem can hence be characterised as asymetric and time dependant.

The aim of the problem is to minimise the overall cost of the travel that can be formulate as follow:

$$\min f = \sum_{d=1}^{n} c_{ij}^{d} \tag{3.2}$$

Subject to:

1. Traveler leaves each area exactly once on each day:

$$\sum_{j \in \text{Airports}} x_{ij}^{d} = 1, \quad \forall i \in \text{Areas}, \forall d \in \{1, 2, \ldots, n\} \tag{3.3}$$

2. Traveler arrives at each area exactly once on each day:

$$\sum_{i \in \text{Airports}} x_{ij}^{d} = 1, \quad \forall j \in \text{Areas}, \forall d \in \{1, 2, \ldots, n\} \tag{3.4}$$

3. Binary decision variable indicating whether the traveler travels from area i to area j on day d:

$$x_{ij}^{d} \in \{0, 1\}, \quad \forall i, j \in \text{Areas}, \forall d \in \{1, 2, \ldots, n\} \tag{3.5}$$

4. Traveler must depart from the starting area on the first day:

$$\sum_{i \in \text{Areas}} x_{0i}^{1} = 1 \tag{3.6}$$

5. Traveler must return to the starting area on the last day:

$$\sum_{j \in \text{Areas}} x_{j0}^{n} = 1 \tag{3.7}$$

6. Traveler visits exactly n airports:

$$\sum_{d=1}^{n} \sum_{i \in \text{Areas}} \sum_{j \in \text{Areas}} x_{ij}^{d} = n \tag{3.8}$$

7. If the traveler goes from airport i to airport j on day d, they must leave from airport j on day $d + 1$:**

$$x_{ij}^{d} = x_{jk}^{d+1}, \quad \forall i, j, k \in \text{Areas}, \forall d \in \{1, 2, \ldots, n - 1\} \tag{3.9}$$

where:

- $x_{ij}^d = 1$ if the traveler travels from Area$_i$ to Area$_j$ on day d, otherwise $x_{ij}^d = 0$.
- c_{ij}^d is the cost of traveling from Area$_i$ to Area$_j$ on day d.

3.2 Instances Description

We are given a set of 14 Instances $I_n = I_1, I_2, ..., I_3, I_4$ that we have to solve. Every instances has the same overall structure.

For example if we take instance I_4, the first few lines are:

```
13 GDN
first
WRO DL1
second
BZG KJ1
third
BXP LB1
```

That means that the Traveller starts from airport GDN, that belongs to the starting area. Then we are given the list of airports that are in every zones. For example the second zone names second have two airports: WRO and DL1.

Each instances named differently the zones but the format is the same. For I_14:

```
fltbhfhwbn
FSE
twykgmcoll
SYY
```

That means that the first zone is fltbhfhwbn and has the airport FSE, twykgmcoll is zone 2 and so on.

. . . LWO KBP 0 515 LWO DOK 0 1032 LWO ODS 0 585 LWO ESB 0 1106 LWO IST 0 880

a	b	c	d
1	2	3	4
5	6	7	

Chapter 4

Solving the 14 instances using a heuristic

(TODO) Describe implementation details

Chapter 5

Results and performance

(TODO) Present the results and discuss any differences between the findings and your initial predictions/hypothesis

(TODO) Interpret your experimental results - do not just present lots of data and expect the reader to understand it. Evaluate what you have achieved against the aims and objectives you outlined in the introduction

Chapter 6

Conclusion

(TODO) Explain what conclusions you have come to as a result of doing this work. Lessons learnt and what would you do different next time. Please summarise the key recommendations at the end of this section, in no more than 5 bullet points.

6.1 Summary of Work

6.2 Critics

6.3 Future Work

(TODO) The References section should include a full list of references. Avoid having a list of web sites. Examiners may mark you down very heavily if your references are mainly web sites.

Chapter 7

Progress and next steps

7.1 Coding

7.1.1 class data preprocessing

Take the instance and create all the needed attributes and methods

7.1.1.1 The class

```
def __init__(self,instance_path):
    self.instance_path=instance_path

    self.info, self.flights = self.read_file(f_name=self.instance_path)

    self.number_of_areas,self.starting_airport=int(self.info[0][0]),self.info[0][1]


    self.flights_by_day_dict = self.flights_by_day(flight_list=self.flights)

    self.flights_by_day_dict=self.redistribute_day_zero(data=self.flights_by_day_dict,number_o


    self.flights_by_day_dict=self.remove_duplicate(flights_by_day=self.flights_by_day_dict)

    self.list_days= self.flights_by_day_dict.keys()
    self.airports_by_area = self.get_airports_by_areas()
    self.area_by_airport=self.invert_dict(original_dict=self.airports_by_area)


    self.starting_area=self.associated_area_to_airport(airport=self.starting_airport)
    self.list_airports=self.get_list_of_airports()
    self.list_areas=list(self.airports_by_area.keys())

    self.areas_connections_by_day=self.possible_flights_from_zone_to_zone_specific_day()
```

7.1.1.2 What needs to be done

- For the big instances, it takes a lot of time to preprocess: solutions are directly eliminated because it runs out of time
- The issue is because: self.redistribute day zero and self.remove duplicate
- Should we improve the time complexity of these 2 functions to avoid this problem? Can we consider we only focus on the execution time of the heuristic algorithm?

7.1.2 class heuristic operators

- Here we only define the classic low level heuristic operator like swap, reverse..

7.1.3 class heuristics

7.1.3.1 The class

```
class heuristics:
def __init__(self, data_preprocessing_class):
    self.data = data_preprocessing_class

    self.starting_airport = self.data.starting_airport
    self.starting_area=self.data.starting_area
    self.total_cost = 0
```

7.1.3.2 Errors and trials

- Tried to solve this problem using graphs: unsucessfull
- I have tried to find LLH: unsucessfull because I did not have all the constraints of the problems
- Works: for each instance I find the whole feasible area solutions: an area solution is considered feasible if all the areas are visited, starting=ending and at least it exists a flight between these areas

7.1.3.3 What needs to be done

Redefine the exact scope of the dissertation: we want solve the KIWI problem using a heuristic. The question is which one. Do we want to compare it with the Local Search results? If yes I'll need the code

What Yaros did: Single point optimisation and RL

Nested problems: first you have to find a feasible area solution and then in this feasible area solution you can find the optimal solution
Problem: I don't really see known names of heuristics on my algo
Question: how can we handle these two operations knowing that when we check if an area solution is feasible we check that there is at least one flight but we can also incorporate to pick the good flight at this change.

What I plan: use some kind of GA.

I think we can try to use a population based algorithm

Comment properly the whole code - I think the structure is fine at the moment

Target: first draft of the dissertation Friday 16th of August -¿ Is it okay for Ahmed?
Goal: all the lecture review written - 80% of the analysis done and my heuristic works - then time to do final comparison

Add section to method - methodology - flow chart

Mimetic algorithm