T.C.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

MINIMUM K-CHINESE POSTMAN PROBLEM

AKİF KARTAL

SUPERVISOR
PROF. DR. DİDEM GÖZÜPEK

GEBZE
2022

T.C.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

MINIMUM K-CHINESE POSTMAN PROBLEM

AKİF KARTAL

SUPERVISOR
PROF. DR. DİDEM GÖZÜPEK

2022
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 16/06/2022 by the following jury.

JURY

Member
(Supervisor) : PROF. DR. DİDEM GÖZÜPEK

Member : YRD.DOÇ.DR. ZAFEİRAKİS ZAFEİRAKOPOULOS

ABSTRACT

In this project, we present a heuristic and exhaustive search algorithm for the minimum k-Chinese postman problem. We considered the minimum k-Chinese postman problem is, given a multigraph $G = (V, E)$ initial vertex $s \in V$ length $l(e) \in N$ for each $e \in E$ the *minimum k-Chinese postman problem* is to find *k* tours(cycles) such that each containing the initial vertex *s* and each edge of the graph has been traversed at least once and the most expensive tour is minimized. This problem is NP-hard and we tried to solve it with a polynomial-time algorithm. For this purpose, we created one polynomial-time algorithm and one exponential-time algorithm and we made a complexity analysis for these algorithms. After creating algorithms we compare them with different parameters by looking at the results and running time. We saw that when the k value is increasing, the heuristic algorithm produces better results in a very short time.

Keywords: heuristic, exhaustive search, NP-hard, polynomial-time, exponential-time, parameters, running time, complexity analysis

ÖZET

Bu projede minimum k-Chinese postman problemi için sezgisel ve kapsamlı arama algoritması sunuyoruz. Minimum k-Chinese postman probleminin tanımı şu şekildedir. Verilen bir multigrafda s adında bir başlangıç noktası vardır ayrıca her 2 nokta arasındaki yol için bir yol uzunluğu vardır. Bu probleme göre bizim amacımız, bu graf üzerinde öyle bir k tane tur ya da dolaşım bulacağız ki her bir turda başlangıç noktası olacak ve graf'taki her bir yoldan en az bir kere geçilmiş olacaktır. Bizim amacımız bu k tane turdaki en büyük uzunluğu sahip turun uzunluğunu en aza indirmektir. Bu problem bir NP-hard problemdir ve biz bu problemi bir polinom zamanlı algoritma ile çözmeye çalıştık. Bu amaç için bir tane polinom zamanlı bir tanede üstel zamanlı olmak üzere iki tane algoritma geliştirdik ve bu algoritmaların karmaşıklık analizini yaptık. Bu algoritmaları oluşturduktan sonra farklı parametrelerle test edip birbirleri ile sonuç ve çalışma süresi bakımından karşılardık ve sonuç olarak gördük ki k değerini artırdıkça çok kısa bir sürede sezgisel algoritma daha iyi sonuçlar buluyor.

Anahtar Kelimeler: sezgisel, kapsamlı arama, NP-hard, polinom zamanlı, üstel zamanlı, çalışma süresi, karmaşıklık analizi

ACKNOWLEDGEMENT

My dear supervisor, who does not spare his interest and support in the planning, research, execution, and formation of this project, whose vast knowledge and experience I have benefited from, I would like to express my eternal and sincere thanks to Didem Gözüpek.

Also, I would like to express my endless and sincere thanks to my teacher, Zafeirakis Zafeirakopoulos who is leading the way for the improvement of this project.

Lastly, I would like to express my respect and love to my family, who supported me in every way during my education, and to all my teachers who set an example for me with their lives.

Akif Kartal

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviation	:	**Explanation**
GUI	:	Graphical User Interface
CPP	:	Chinese Postman Problem
MIN	:	Minimum
MAX	:	Maximum

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

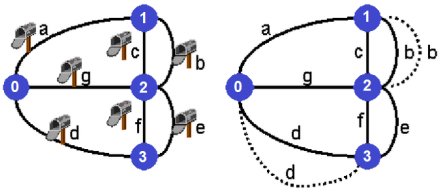
A graph is a data structure composed of a set of objects (nodes) equipped with connections (edges) among them. Graphs can be directed if the connections are oriented from one node to another (e.g. Alice owes money to Bob), or undirected if the orientation is irrelevant and the connections just represent relationships (e.g. Alice and Bob are friends). A graph is said to be complete if all nodes are connected to each other. A directed graph with no loops is said to be acyclic. A few practical examples of graphs are friendship networks (e.g. on social media), genealogical (family) trees, molecules, particles produced at the Large Hadron Collider, and a company's organizational chart.[1]
To find the most efficient way of traversing an entire graph is a widely spread problem in today's society. For a snow truck to plow all snow on every street in a town in the minimal consumed time is only one of a vast amount of applications of this problem. Finding solutions for this problems would lead to both a financial and an environmental improvement to companies and cities all over the world.[2]

1.1. Project Definition

In this project, We tried to solve Minimum k-Chinese Postman Problem which is given a multigraph $G = (V, E)$ initial vertex $s \in V$ length $l(e) \in N$ for each $e \in E$ the *minimum k-Chinese postman problem* is to find *k* tours(cycles) such that each containing the initial vertex *s* and each edge of the graph has been traversed at least once and the most expensive tour is minimized.
To solve this problem, we have implemented 2 different algorithms and evaluated them in different perspectives.

The problem has following inputs.

- *s*, initial vertex
- *k*, given positive number
- *l(e)*, length for each edge
- *n*, number of vertices(nodes)
- *e*, number of edges



(a) Simple Multigraph Example

(b) Postmans

1.2. The Goal of the Project

Making reason for this project is to implement a heuristic algorithm for the Minimum k-Chinese Postman Problem from Literature.
In Literature, there are a reasonable number of algorithms for this problem but finding implemented one is nearly impossible because these types of algorithms are NP-hard.
Also, People can see how to solve these types of problems, how to make a complexity analysis, and how to make a comparison between different algorithms.
Lastly, after publishing this project people can use these solutions and improve them.

2. Project Design and Details

In order to make this project, we need to consider and determine details about this project.

2.1. Project Design Plan

In the following image you can see the project design plan in a good manner.

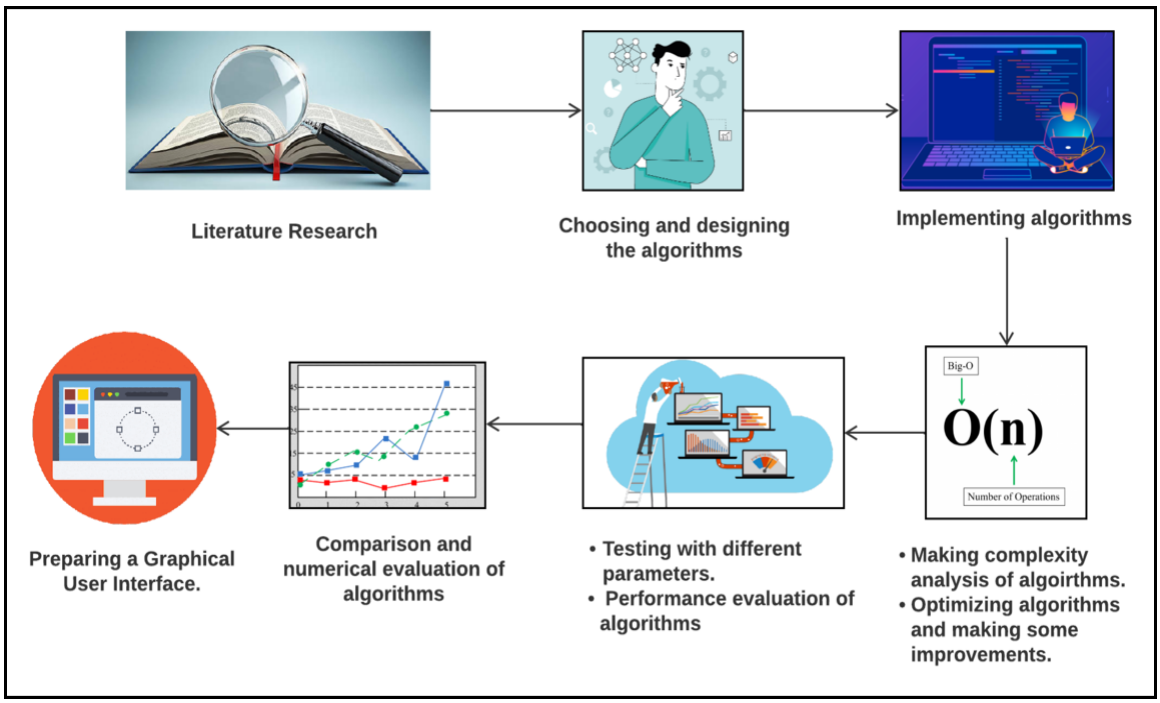


Figure 2.1: Project Design Plan

2.2. Project Requirements

- Making literature research and understanding the problem.
- Choosing and designing algorithms.
- Implementing both heuristic and exhausted search algorithms.
- Making complexity analysis of the algorithms.
- Testing with different parameters and performance evaluation of algorithms.

- Comparison and numerical evaluation of algorithms.
- Showing the comparison average results on the charts.
- Preparing a GUI and running algorithm on that GUI.

2.2.1. Literature Research

In order to solve this problem we have to make a deep research and reading in literature. For Minimum k-CPP, I have read following articles.

Author	Article
Dino Ahr, Gerhard Reinelt	New heuristics and lower bounds for the min-max k-chinese postman problem[3]
Kaj Holmberg	Heuristics for the weighted k-Chinese/rural postman problem with a hint of fixed costs with applications to urban snow removal[4]
G. Gutin, G. Muciaccia	Parameterized Complexity of k-Chinese Postman Problem[5]
Anton Hölscher	A Cycle-Trade Heuristic for the Weighted k-Chinese Postman Problem [2]
Dino Ahr, Gerhard Reinelt	A tabu search algorithm for the min–max k-Chinese postman problem[6]

Table 2.1: Found and Read Articles while Researching

2.2.2. Tools and Technologies

In order to make this project following tools and technologies have been used.

Figure 2.2: Tools and Technologies

1. **Python 3.9:** This is used as as programming language to implement algorithms and gui.
2. **Windows 10:** This is used as operating system.
3. **PyCharm IDE:** This is used to as development environment.
4. **igraph:** This python library is used to generate and draw graphs.
5. **PyQt5, Qt Designer:** These are used to make graphical user interface.
6. **Git and Github:** These are used to keep source code.

3. Heuristic Algorithm

In this module, I tried to determine whether a dormitory comment is positive or negative. In order to do this, I have to use some Natural Language Processing methods. In my solution, in order to create word vectors, I have used TF-IDF(term frequency-inverse document frequency) vectorizer and Logistic Regression as a Classifier from python scikit-learn library. In order to complete this module following steps have been applied.

1. Find a good dataset
2. Clear data and remove noise in the dataset
3. Split train and test data
4. Create and train a model by using TF-IDF vectorizer and Logistic Regression
5. Test that model accuracy
6. Export and deploy the model

Next we will see these steps in detailed way.

3.1. Algorithm Steps

As we know our comments will be in Turkish. But there was no good dataset in Turkish for dormitory comments all we have are movie reviews or product reviews in Turkish. Therefore while I was searching for a good dataset I have found the this dataset.[1] In this English dataset, we have 35038 hotel comments dataset with the response(positive or negative). For this module, I will use this dataset by translating Turkish comments into English.

3.2. Algorithm implementation Details

After finding the dataset, I can create and train our model. Before that, I need to clear and remove noise in the data. In the following code pieces, we can see these steps.

[1]https://www.kaggle.com/anu0012/hotel-review?select=train.csv

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

tvec = TfidfVectorizer(ngram_range=(1, 1))
clf2 = LogisticRegression()

model = Pipeline([('vectorizer',tvec)
                  ,('classifier',clf2)])

model.fit(attribute_train, target_train)
```

Listing 3.1: Creating Model

In following code piece, I will test my model by using Confusion Matrix. In this test my test data will be %10 of train data. For more information check chapter **??**.

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

verdict = model.predict(attribute_test)
confusion_matrix(verdict, target_test)

print("Accuracy : ", accuracy_score(verdict, target_test))
```

Listing 3.2: Testing Model with Confusion Matrix

3.2.1. Sorting the edges

In order to use this model in my mobile application I need deploy it to the public so that my application can use it. In order to do this first I need to export my model into a file by using python **pickle**[1] library.

```
import pickle
path='/content/drive/MyDrive/Colab Notebooks/model2/'
pickle.dump(model, open(path+'model.pkl','wb'))
```

Listing 3.3: Exporting Model Using Pickle into a File

[1]https://docs.python.org/3/library/pickle.html

After exporting our model I can use that model in our API. In the following code piece we can see how I am using the exported model file to integrate with my mobile application.

3.2.2. Creating the Closed Walks

```
import pickle
from flask import Flask
from flask import jsonify
from flask import request
from googletrans import

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))
translator = Translator()

@app.route("/api/v1/getResult", methods=['POST'])
def getresult():
    request_data = request.get_json()
    comment = request_data['comment']
    com_en = translator.translate(comment, dest="en").text
    com_en = com_en.replace("country", "dormitory")
    com_en = [com_en]
    result = model.predict(com_en)
    return jsonify({'result': result[0]})
```

Listing 3.4: Use of Exported Model to Evaluate Translated Comment in API

3.2.3. Adding Dummy Tours

3.2.4. Merging Tours

4. Exhaustive Search Algorithm

In order to store data, we need a database in this project. But the important thing is this database should be **reachable** from anywhere which means it shouldn't be a local database on any computer. Therefore cloud databases is good for this project. In-state of the art **Google firebase platform** is mostly used for this purpose because it has powerful options and is free for the starter package. Firebase works with two different databases. The Real-Time Database is the original Firebase database product, and Cloud Firestore is a new and improved version of the Real-Time Database.[1] In this project, we have used the Cloud Firestore database of Firebase.

4.1. Algorithm Steps

One of the requirements of this project collecting data about dormitories that are around Gebze Technical University. In order to do this, I have collected data by calling each dormitory and talking with the responsible person about that dormitory. In this process, **some of the people avoided giving information about that dormitory.** Therefore there can be some missing and wrong information about dormitories.

4.2. Algorithm implementation Details

Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps at a global scale.[2]

4.2.1. Finding All Cycles

Below we have the tables and relations between them in the figure 4.1. As a relation, we have one-to-one and one-to-many relations between tables. Also in tables, most data types are the **string to manage** data easily in the application.

We can see real data in the figure 4.2. On the left side, we have 4 database tables on the center, we have each separate data in table with its unique id on the right side, we have the context of the data.

[1] https://blog.back4app.com/firebase-vs-firestore/
[2] https://firebase.google.com/products/firestore

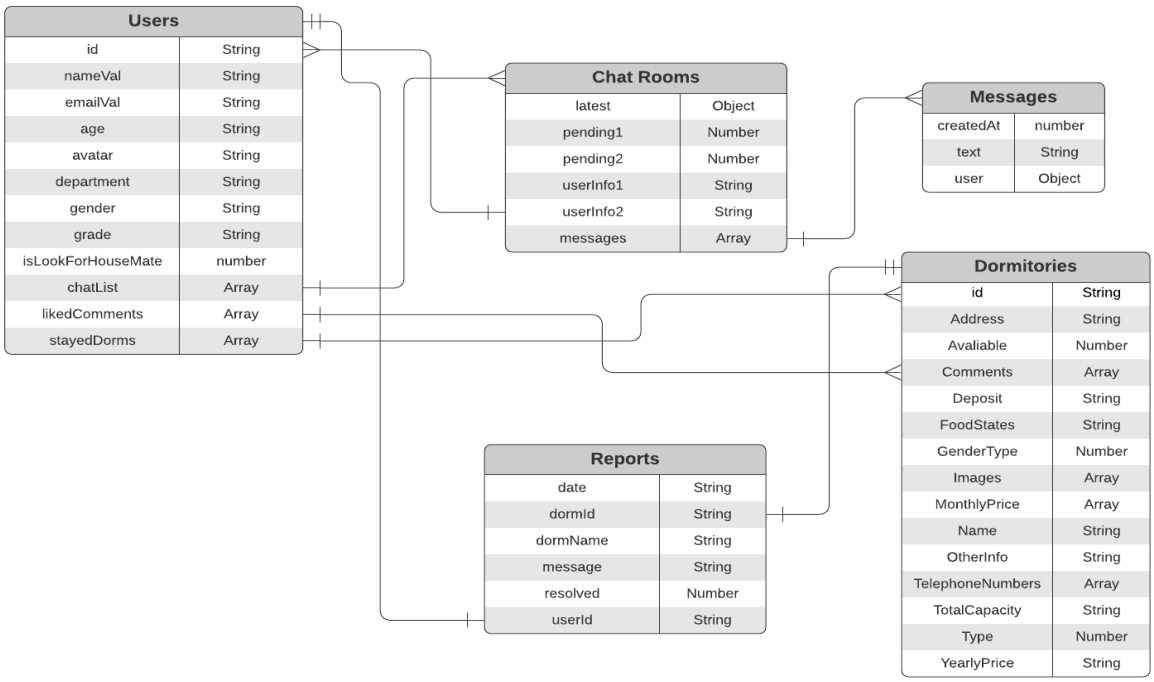


Figure 4.1: Table Relations in Database

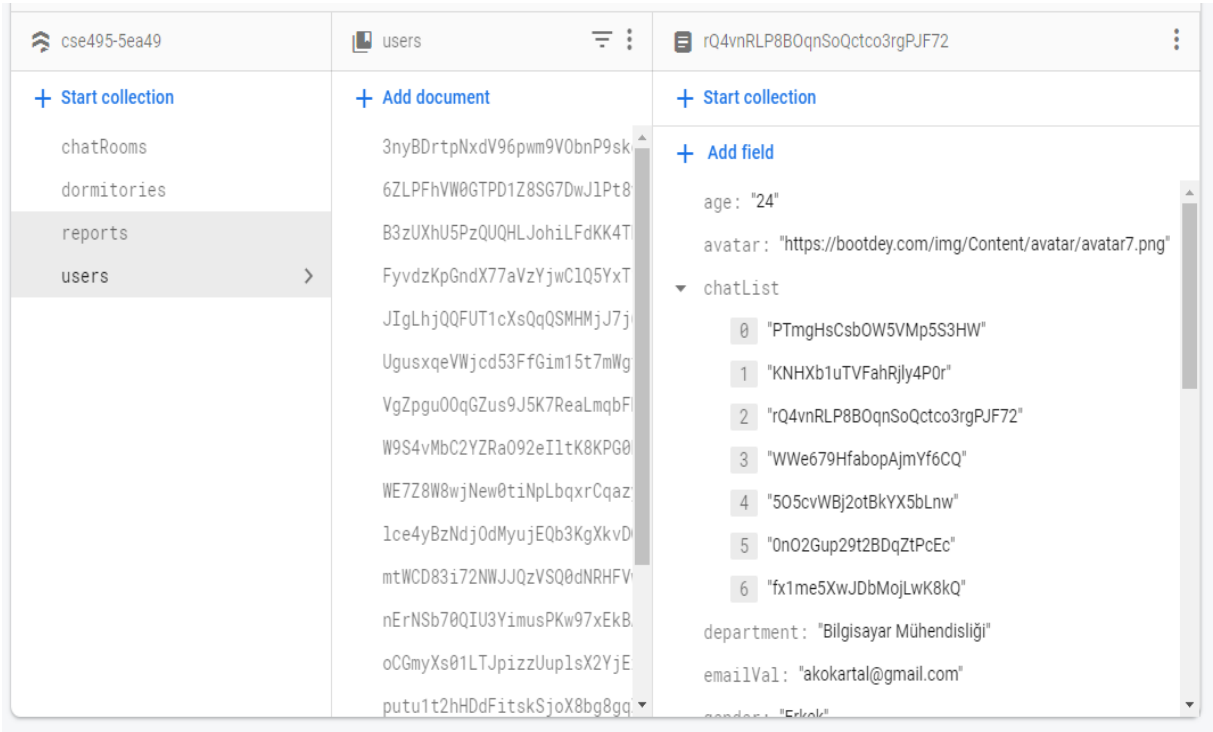


Figure 4.2: Real Data and Tables in Firestore Database

4.2.2. Finding All k Combinations

4.2.3. Finding All Proper Cycles

4.2.4. Choosing the Optimal Cycle

5. Complexity Analysis of Algorithms

Making the mobile application is the most involved part of this project and to make mobile application I have used react native technology which is one of the heavily used technology in business.
One of the critical part of react native is **you need to make everything with code** which means **there is no drag and drop** feature for the components such as buttons etc.

Figure 5.1: Mobile Application Logo

5.1. Complexity Analysis of Heuristic Algorithm

Next, we will see each screen and features of the application one by one.

5.2. Complexity Analysis of Exhaustive Search Alg.

6. Performance Evaluation of Algorithms

In this part, since exhaustive search algorithm can only run limited data sizes, we will evaluate the performance of the two algorithms separately.

6.1. Performance Evaluation of Heuristic Algorithm

To test the heuristic algorithm, we will create **50 different graphs** and we get the average with different parameters.

6.1.1. Changing Node Count

In this test, we will increase the node count and we will see the running time of the algorithm. The graph density will be constant. Parameters will be as follows.

- **initial vertex** = 0
- **k** = 20
- **number of nodes** = 8, 10, 12, 14, 16
- **number of edges** = $((n * (n - 1))/2) - 5$

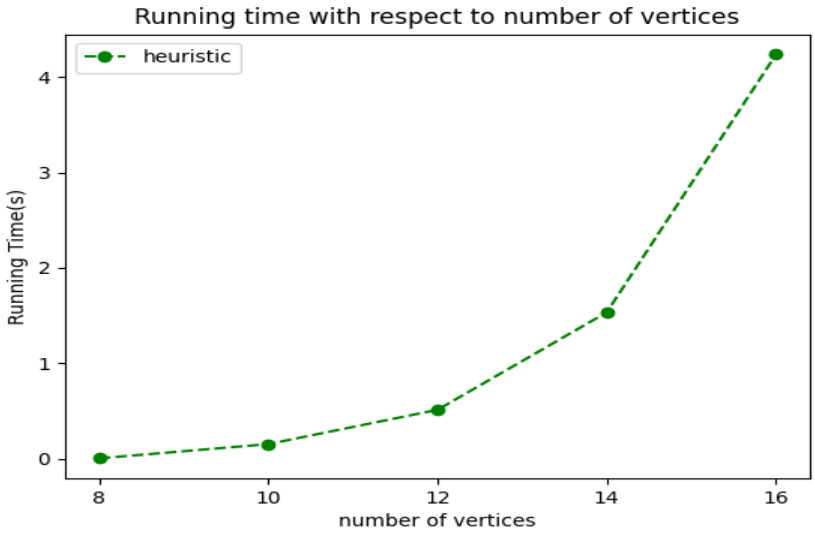


Figure 6.1: Running time with respect to number of vertices

In the above chart, you can see that while node count is increasing, the running time increases in the heuristic algorithm. This is happening because the k value is constant which means when the graph is increasing we have more cycles than 20 such as 35, therefore, **we need to merge some cycles** and this operation takes time because it is the worst case for our heuristic algorithm.

6.1.2. Changing Node Count and k Value

In this test, we will increase the node count as well as the k value according to node count and we will see the running time of the algorithm. The graph density will be constant. Parameters will be as follows.

- **initial vertex** = 0
- **number of nodes** = 8, 10, 12, 14, 16
- **number of edges** = $((n * (n - 1))/2) - 5$
- **k** = $n * 5$

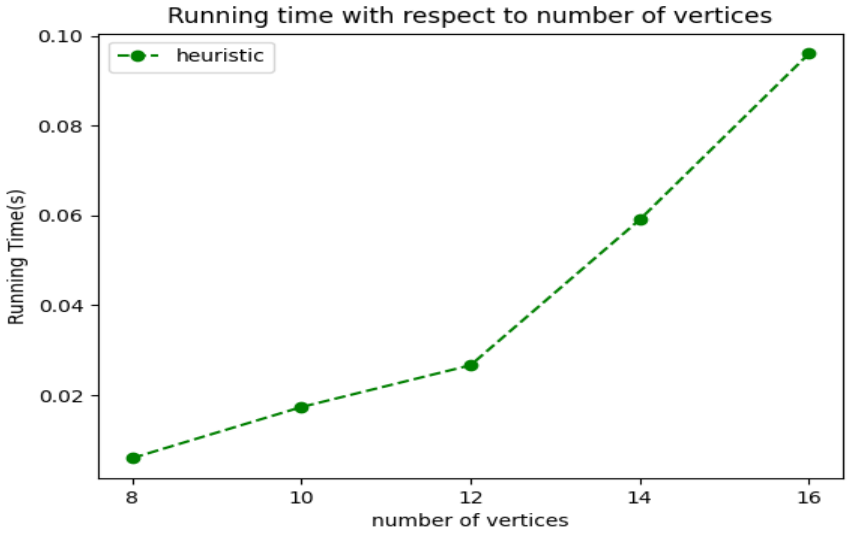


Figure 6.2: Running time with respect to number of vertices and k value

As you can see in the chart, still running time is increasing but in this case, times are very small. In the previous test, we have more than 4 seconds but now we have 0.09 seconds. This happens because the **k value is proportional with graph size** therefore, we don't have to merge tours and we gain time. **This means on the same graph if the k value is increased, taken time will decrease in the heuristic algorithm.**

6.2. Performance Evaluation of Exhaustive Search Alg.

6.2.1. Changing Node Count

In this test, we will increase the node count and we will see the running time of the algorithm. The graph density will be constant. Parameters will be as follows.

- **initial vertex** = 0
- **k** = 4
- **number of nodes** = 4, 5, 6
- **number of edges** = 5, 8, 10

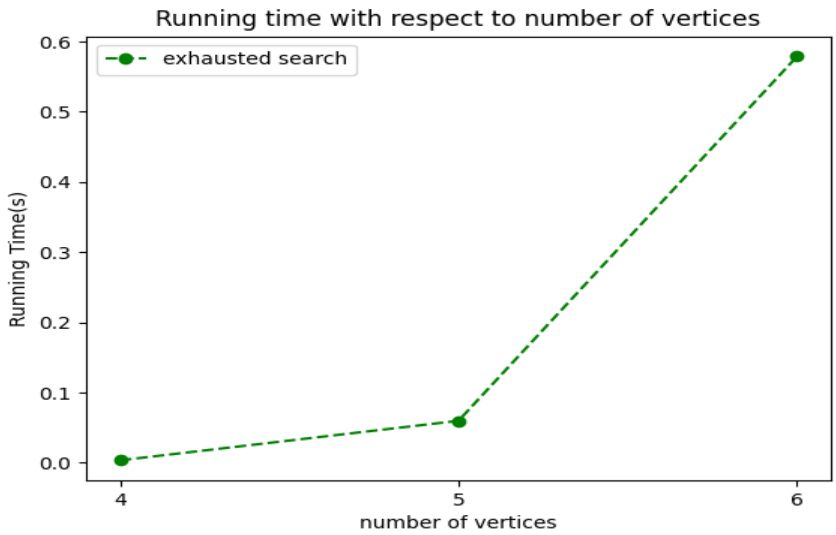


Figure 6.3: Running time with respect to number of vertices

In the above chart, you can see that while node count is increasing, the running time increases in the exhausted search algorithm. As you can see in the chart running time is small according to the exhausted search. This happens because the **k value is constant and small** also the graph size is small.

6.2.2. Changing Node Count and k Value

In this test, we will increase the node count as well as the k value according to node count and we will see the running time of the algorithm. The graph density will be constant. Parameters will be as follows.

- **initial vertex** = 0
- **number of nodes** = 4, 5, 6
- **number of edges** = 5, 8, 10
- **k** = 3, 5, 7

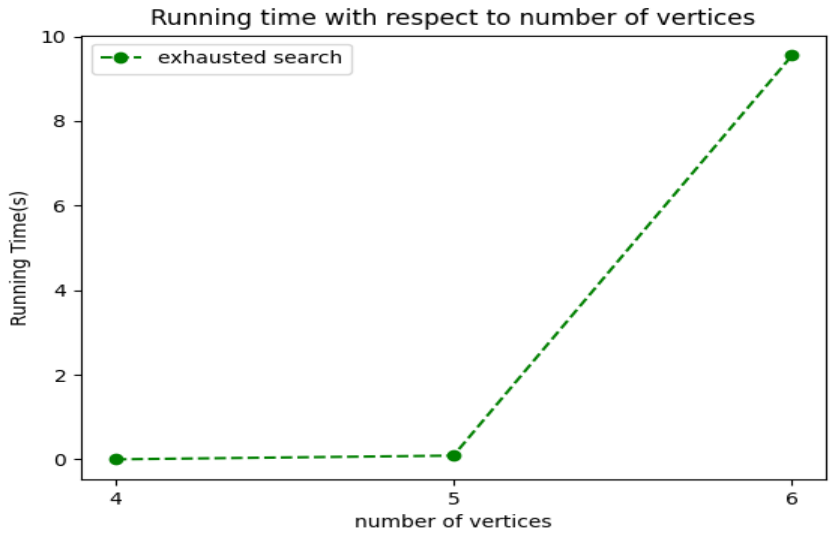


Figure 6.4: Running time with respect to number of vertices and k value

As you can see in the chart, still running time is increasing but in this case, times are very big. In the previous test, we have less than 1 second but now we have almost 10 seconds. This happens because the **k value is proportional with graph size and it gets a bigger value** this means on the same graph **if the k value is increased, taken time will increase in the exhausted search algorithm.**

7. Comparison and Numerical Evaluation

In this part, we will compare the two algorithms for running time and maximum length of cycles. Also, we will make a numerical evaluation of the results.

In order to make numerical evaluation we have following test cases.

- We will change the both number of nodes and the number of edges which means we will have a bigger graph. Also, the k value will be constant.
- We will change only the number of edges which means the density of the graph will change. Also, the k value and number of nodes will be constant.
- Lastly, we will change only the k value. Also, number of node and number of edge will be constant.

7.1. Test Case 1

In this test, we will change the both number of nodes and the number of edges which means we will have a bigger graph. The k value will be constant.
Parameters will be as follows.

- **initial vertex** = 0
- **k** = 3
- **number of nodes** = 4, 5, 6, 7, 8
- **number of edges** = 6,9,10,12,12

7.1.1. Running Time Comparison

Figure 7.1: Running time with respect to number of vertices

In the above chart when the graph is growing running time of the exhausted search algorithm is increasing exponentially. This is an expected result because in bigger graphs we have more cycles and to get k combination of that cycles we need more time.

7.1.2. Maximum Length Comparison

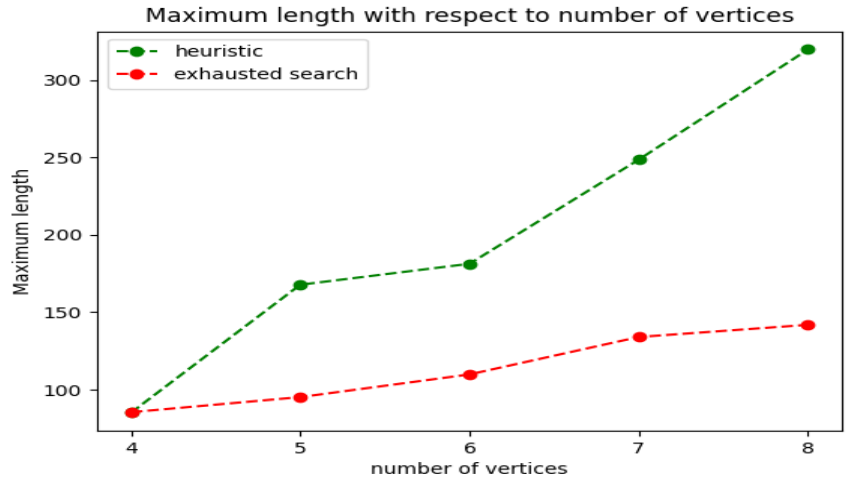


Figure 7.2: Maximum Length with respect to number of vertices

In the above chart when the graph is growing and the k value is small and constant heuristic algorithm gets worse results. This is an expected result because when the k value is small we have to merge found tours after merging maximum length is increasing. For example, if k value 3 then let's say we get 10 cycles in heuristic in order reduce 10 to 3 we have merge after merging maximum length is increasing therefore exhausted search gets better results.

7.2. Test Case 2

In this test, we will change only the number of edges which means the density of the graph will change. The k value and number of nodes will be constant.
Parameters will be as follows.

- **initial vertex** = 0
- **k** = 4
- **number of nodes** = 6
- **number of edges** = 7, 8, 9, 10, 11

7.2.1. Running Time Comparison

Figure 7.3: Running time with respect to number of edges

In the above chart when the graph density is growing running time of the exhausted search algorithm is increasing exponentially. This is again an expected result because in the dense graphs we have more cycles and to get the k combination of that cycles we need more time.

7.2.2. Maximum Length Comparison

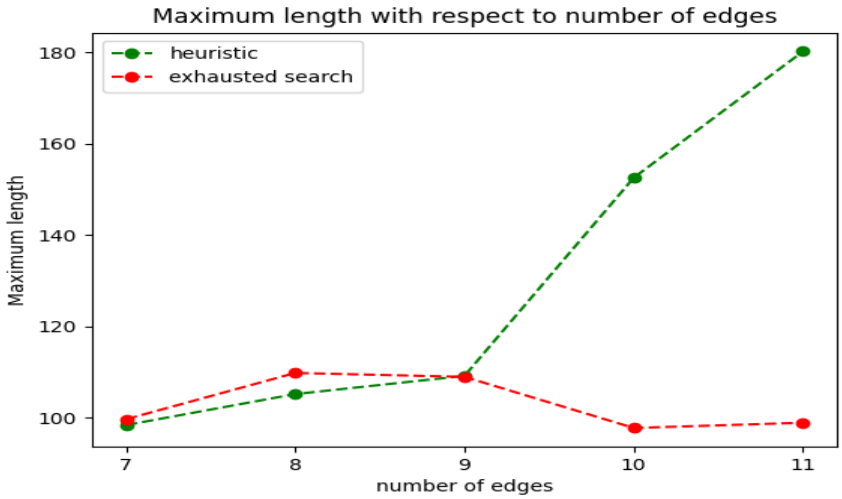


Figure 7.4: Maximum Length with respect to number of edges

In the above chart when the graph density is growing for the points in which the graph is not dense, the heuristic algorithm gets a better result. But in dense graphs, the heuristic algorithm gets a worse result. This is again an expected result because in the dense graphs heuristic algorithm produce more cycle and it has to merge them after merging operation result is getting an increase.

7.3. Test Case 3

In this test, we will change only the k value. Number of node and number of edge will be constant.
Parameters will be as follows.

- **initial vertex** = 0
- **k** = 4, 5, 6, 7, 8
- **number of nodes** = 6
- **number of edges** = 10

7.3.1. Running Time Comparison

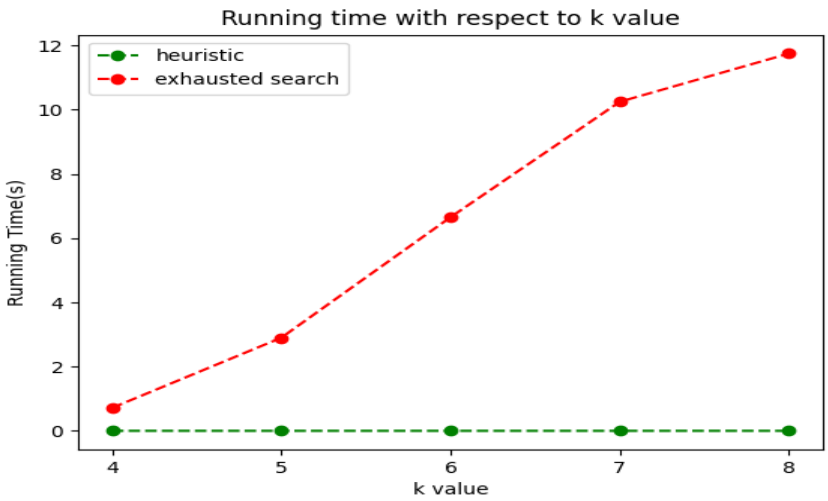


Figure 7.5: Running time with respect to number of edges

In the above chart when the k value is increasing running time of the exhausted search algorithm is increasing. This is again an expected result because the exhausted search algorithm has to get the k combination in any case and this operation takes time.

7.3.2. Maximum Length Comparison

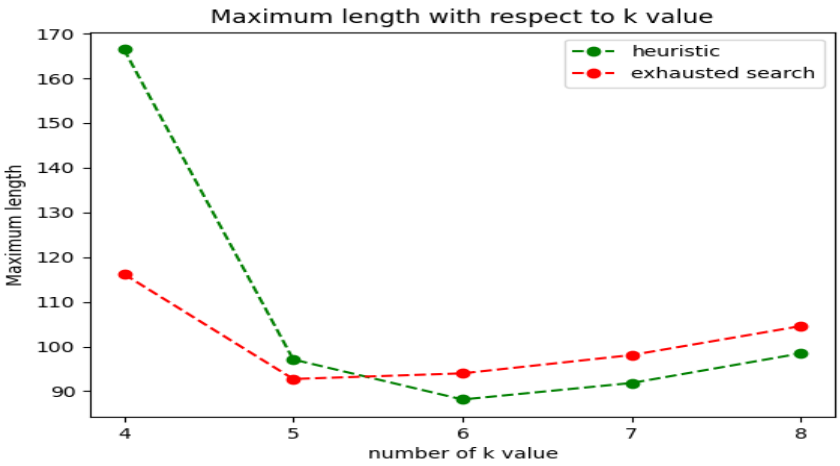


Figure 7.6: Maximum Length with respect to number of edges

In the above chart when the k value is increasing heuristic algorithm gets a better result. This is again an expected result because when the k value is big heuristic algorithm doesn't need to make a merging operation therefore it produces a better result as we expect in this project.

7.4. Summary of Comparisons

As we have seen in the above comparisons unfortunately in any case exhausted search takes a long time. But still, it can produce very good results. On the other hand heuristic algorithm is good for running time. But it produces no good result when the k value is small. Because in that case, it has to make merge operation after merging the result is not good according to the exhausted search algorithm that means we can optimize the merging operating. But when the k value is big it produces good results in a very small time.

8. Graphical User Interface

Making the mobile application is the most involved part of this project and to make mobile application I have used react native technology which is one of the heavily used technology in business.
One of the critical part of react native is **you need to make everything with code** which means **there is no drag and drop** feature for the components such as buttons etc.

Figure 8.1: Mobile Application Logo

8.1. User Interface Design and Results

Next, we will see each screen and features of the application one by one.

9. Success Criteria

For this project, we have determined 4 success criteria. These are;

1. Heuristic algorithm complexity will be better than $\mathcal{O}(|E|^4)$
2. Creating 50 different random graphs for each test case in performance testing and creating 20 different random graphs for each comparison case and taking the average of them.
3. Getting results with the heuristic algorithm in less than 1 second when number of nodes < 25 and *k* is not constant.
4. When *k* is big and proportional to the number of edges, the results of the heuristic algorithm are $\%10$ better than the exhaustive search.

Next, we will see that how I have accomplished these criteria one by one.

9.1. Criterion 1

- Heuristic algorithm complexity will be better than $\mathcal{O}(|E|^4)$

I have accomplished this criterion successfully. In the mobile application, we have **14 different** screens. These are start, login, sign up, forgot password, dormitory list, housemate list, user information, chat, chat list, user profile, error reports list, dormitory information, error reporting, and dormitory comments. For more information check chapter 8.1.

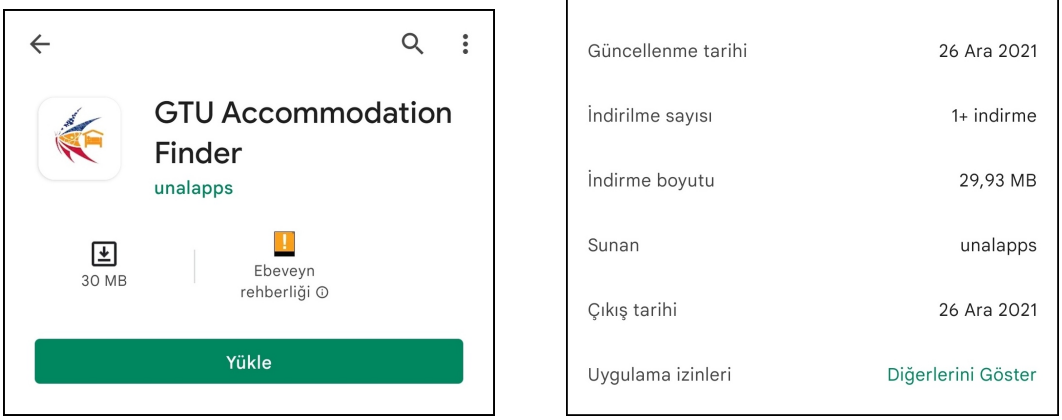
9.2. Criterion 2

- Creating 50 different random graphs for each test case in performance testing and creating 20 different random graphs for each comparison case and taking the average of them.

I have accomplished this criterion successfully. In order to show this, I have uploaded my mobile application on the google play store.[1] In the google play store, **my application download size is 29,93 MB**.

[1] https://play.google.com/store/apps/details?id=com.cse495.GTUAccommodationFinder

Check the application size in the following images.



(a) Mobile Application on Google Play

(b) Mobile Application Details on Google Play

Figure 9.1: Details about Application Size in Google Play Store

In following graph, which is taken from google play console[1] we can see my total application size. according to Google play reports my total application size is **33.7 MB.**

9.3. Criterion 3

- Getting results with the heuristic algorithm in less than 1 second when number of nodes < 25 and k is not constant.

I have accomplished this criterion successfully. In order to show this, I tested my model with 2 different test method. Below we will see this methods one by one.

9.4. Criterion 3

- When k is big and proportional to the number of edges, the results of the heuristic algorithm are $\%10$ better than the exhaustive search.

I have accomplished this criterion successfully. In order to show this, I tested my model with 2 different test method. Below we will see this methods one by one.

[1]https://play.google.com/console/about/

9.5. Summary of Success Criteria Results

In the following table, we can see the all success criteria's expected and actual results. Note that for each success criterion we got the expected results and we are successful but in the heuristic algorithm **merging tours steps is open to improvement.**

Success Criterion	Expected	Actual	Result
Heuristic algorithm complexity will be better than $\mathcal{O}(\|E\|^4)$	$\mathcal{O}(\|E\|^4)$	$\mathcal{O}(\|E\|^3)$	Successful
Creating 50 different random graphs for each test case in performance testing and creating 20 different random graphs for each comparison case and taking the average of them.	50 and 20 different random graphs for each test case	50 and 20 different random graphs for each test case	Successful
Getting results with the heuristic algorithm in less than 1 second when number of nodes < 25 and k is not constant.	in less than 1 second	0.10 second	Successful
When k is big and proportional to the number of edges, the results of the heuristic algorithm are %10 better than the exhaustive search.	%10 better	%10 better	Successful

Table 9.1: Success Summary of All Results

10. Conclusions

In this project, I have tried to solve and make it easy to find a place to stay while studying in university for GTU Students. As we have seen this problem is mostly a software engineering problem, therefore, I had to take action according to this and I did mostly.
In order to solve this problem I have applied the following steps;

1. Define Requirements
2. Make a design(both visual and architectural)
3. Divide design into modules
4. Code modules one by one
5. Test each module
6. Combine modules
7. Deploy application
8. Maintenance application

As an engineer in the design step, I have considered both visual design and architectural design. My visual design is user-friendly and meets the requirements as expected. On the other hand in the architectural design step, I have chosen the most appropriate technologies for the project and heavily used ones. For example, in order to integrate the sentiment analysis module with this project, I have used a request-response mechanism with HTTP protocol by using python flask technology.

By dividing the big project into modules I have conquered each small piece so that my problems were small to solve and test. After finishing the modules I have created this project.

As a result, while making such a project the key point is being agile about both changes and learning technologies since time is limited and you can't avoid changes.

BIBLIOGRAPHY

[1] https://towardsdatascience.com/quick-guide-to-graph-traversal-analysis-1d510a5d05b5.

[2] A. Hölscher, *A cycle-trade heuristic for the weighted k-chinese postman problem*, 2018.

[3] D. Ahr and G. Reinelt, *New heuristics and lower bounds for the min-max k-chinese postman problem*, 2002.

[4] K. Holmberg, *Heuristics for the weighted k-rural postman problem with applications to urban snow removal*, 2015.

[5] A. Y. Gregory Gutin Gabriele Muciaccia, *Parameterized complexity of k-chinese postman problem*, 2014.

[6] D. Ahr, *A tabu search algorithm for the min–max k-chinese postman problem*, 2005.