C

Singapore Institute of Technology - University of Glasgow
Joint Degree in Computing Science Degree Programme

CSC3001 Capstone Project

Please complete the following form and attach it to the Capstone Report submitted.

Capstone Period: **29 Aug 2022** **to** **14 Jul 2023**
Assessment Trimester: **Final Trimester**
Project Type: **Industry**

Work Supervisor Details & Declaration:

Name: William Ng
Designation: Chief Operations Officer (COO)
Department: Operations
Email Address: William@uparcel.sg
Contact Number: 96871808

Academic Supervisor Details:

Name: Lawrence Seow Chee Kiat
Designation: Assistant Professor
Email Address: cheekiat.seow@glasgow.ac.uk
Contact Number: 69086043

Student Particulars & Declaration:

Name of Student: Lim Jun Xian
Student ID: 2000905

I hereby acknowledge that I have engaged and discussed with my **Academic Supervisor** and **Work Supervisor** on the contents of this Final Capstone Report and have sought approval to release the report to the Singapore Institute of Technology and the University of Glasgow.

Signature 

Date: 14 Jul 2023

END OF FORM

Singapore Institute of Technology - University of Glasgow
Joint Degree in Computing Science Degree Programme

Final Capstone Report "Spectral Clustering-Based Approach for Delivery Job Matching"

For **Final Trimester** from 02 May 2023 to 14 July 2023

Lim Jun Xian
Student ID: 2000905

Academic Supervisor: *Lawrence Seow Chee Kiat*

Submitted as part of the requirement for CSC3001 Capstone Project

Table of Contents

List of Figures

Acknowledgments

The completion of this project would certainly not have been possible without the guidance and assistance of many others. In this acknowledgment, I would like to express my gratitude to firstly my work supervisor, William, for his invaluable guidance, patience, encouragement, and committed assistance during my capstone project. Secondly, I would also like to extend my appreciation to my academic supervisor, Professor Lawrence for his constructive recommendations during the preliminary stages of this project which led me toward the right direction in my research.

1. Introduction

1.1. Problem Formulation

1.1.1. Background

A Logistic Service Provider (LSP) is defined as a company that offers supply chain management services including warehousing, transportation, or distribution services [1].

Delivery delays pose a significant challenge for LSPs [2] [3], particularly during peak seasons when meeting delivery deadlines becomes increasingly difficult. These delays primarily stem from two factors: the increasing shortages of Delivery Driver (DD) [4] and inefficiencies in the delivery routings employed by certain LSPs [5] [6].

To address the DD shortage, many LSPs have turned to Crowdsourcing as a solution, enabling them to effectively manage fluctuating demands during peak and off-peak seasons [7]. High-demand periods, such as the Mooncake Festival, Singles Day (11.11), or Christmas, require a larger workforce of DDs, while off-peak seasons necessitate fewer DDs.

Crowdsourcing, at first glance, seems like an optimal solution, as it provides flexibility by engaging Independent Contractors (ICs) according to demand, rather than maintaining a large full-time workforce year-round. This scalability allows LSPs to optimize operations and reduce costs accordingly.

However, relying solely on Crowdsourcing to address the DD shortage during high-demand seasons may not entirely resolve the issue. As more LSPs adopt Crowdsourcing and engage ICs, these ICs, who are not official employees of any specific LSP [8], may register on multiple LSP platforms to access more job opportunities and choose the most lucrative assignments [9].

This situation can lead to an overestimation of available staffing numbers for each LSP. For instance, figure 1 shows an example if Company A claims to have 30,000 drivers, Company B claims to have 20,000 drivers,

and Company C claims to have 10,000 drivers, the total number of drivers across platforms would theoretically be 60,000. However, in reality, some drivers may work simultaneously across multiple platforms to secure more delivery jobs and increase their income. As a result, the actual total number of drivers across platforms could be significantly lower than the sum of the claimed numbers.

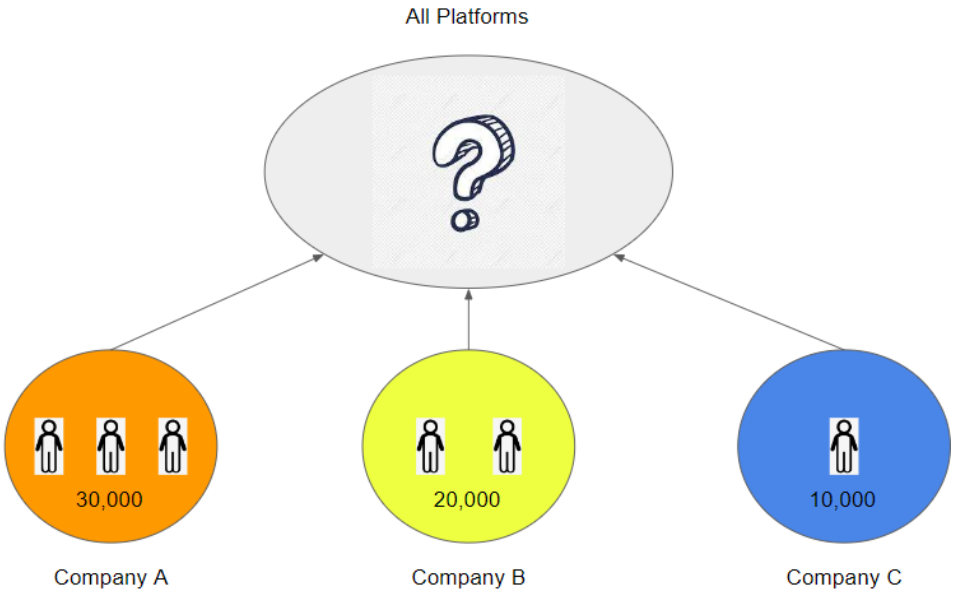

Fig 1. Staffing Numbers Illustration

During peak seasons, this becomes problematic as the number of contractors may be insufficient to meet the overwhelming demands of each LSP. In such situations, LSPs may attempt to outsource their orders to third-party LSPs, such as Company C outsourcing to Company A. However, this outsourcing strategy often fails due to other third-party LSPs also grappling with their own high-demand periods and inflated staffing numbers. Consequently, these third-party LSPs may lack the additional DDs necessary to fulfill external orders.

Thus, the widespread adoption of Crowdsourcing does not effectively address the issue of the DD shortage.

To tackle the second part of the issue, which involves the inefficiency in delivery routings, many LSPs often turn to Route Optimization Software as

a solution. However, it is important to recognize that this approach may not fully address the underlying problem. Despite the capabilities of route optimization software, if an LSP deals with jobs that are geographically far apart, it can still pose limitations on achieving truly optimized routes [10].

Hence, this also substantiates the notion that the commonly used Route Optimization Software may not effectively solve the issue of inefficiency in LSP's delivery routings.

1.1.2. Purpose of Capstone Project

Therefore, a better and more reliable solution to the whole delivery delays issueis required. Hence, the purpose of the Capstone Project is to look to solve the problem by both improving the efficiency of the LSPs' deliveries through optimizing their delivery routes and allowing LSPs to tap on other LSP's or company's idle DDs.

1.1.3. Problem Definition

As mentioned, the delivery delays issue can be divided into two sub-problems, the increasing DD shortage and the inefficiency in some of the LSP's delivery routings.

The increasing DD shortage reduces the number of potential DDs taking up the LSPs' delivery jobs, hence leading to delivery delays when there are not enough DDs to meet the delivery demands.

While the inefficiency in LSP's delivery routings causes more miles and time to be used than needed for each delivery [11], thus leading to delivery delays as well.

Both problems are worthwhile working on because thus far there are yet to be any software solutions that were able to resolve the DD shortage problem directly and efficiently while being able to do route optimization at the same time.

Hence, by solving the two sub-problems of increasing DD shortage and inefficiency in LSP's delivery routings, the delivery delays issue will also be

solved.

1.2. Project Objectives/Project Specifications

1.2.1. Objectives

To address the aforementioned sub-problems, this paper introduces the concept of a Delivery Job Exchange (DJE), which enables companies to engage in job trading or exchanging. The DJE will incorporate two algorithms: an Inbound Algorithm and an Outbound Algorithm. The Inbound Algorithm recommends company administrators to drop undesirable jobs from their existing list of delivery jobs (private jobs) into the DJE job pool (public jobs) to enhance overall delivery efficiency. For instance, it may suggest dropping outlier jobs. On the other hand, the Outbound Algorithm recommends company administrators to consider jobs from the DJE job pool (public jobs) that align well with their current delivery routes. This may involve recommending jobs located near the company's existing jobs (private jobs). Figure 2 provides an illustration of the job exchange flow, including the Inbound and Outbound Job Recommender Algorithms.

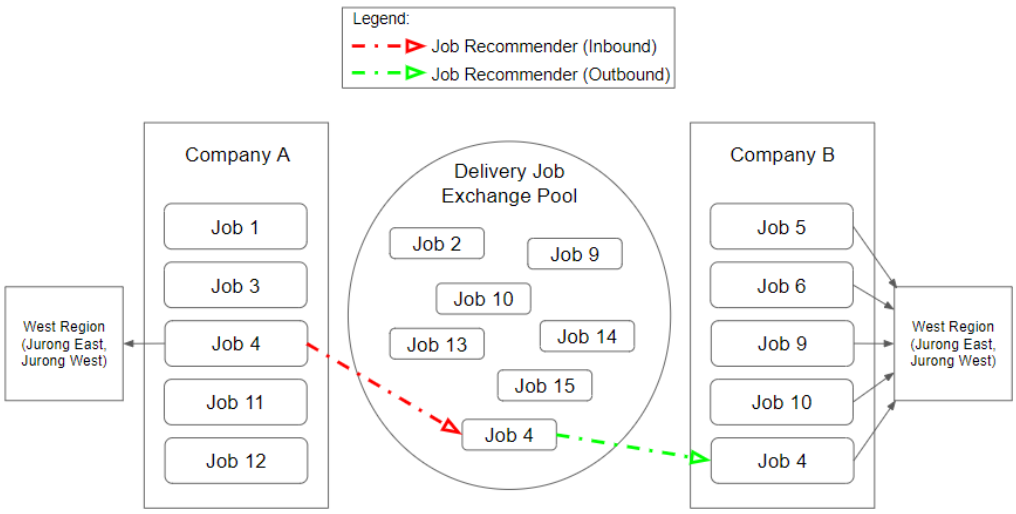

Fig 2. Job Exchange Flow

By implementing this concept, LSPs can effectively address the DD shortage issue by leveraging idle DDs from other companies, regardless of industry, as long as they have joined the DJE. Additionally, the concept improves the delivery routing efficiency of all participating LSPs by recommending the transfer of outlier jobs to the DJE pool and the

acceptance of jobs that are closely related or compatible with their current delivery routes. This comprehensive approach enhances the overall delivery routing efficiency for all participants.

However, given the extensive scope of this concept, this paper will primarily concentrate on the implementation of the Outbound Job Recommender Algorithm. Therefore, the project objective focuses on the implementation of this specific algorithm.

1.2.2. Specifications

- Data Sources: Mockup data of Pending Delivery Jobs (public jobs) & LSPs Jobs (private jobs).
- Machine Learning Algorithm: Spectral Clustering.
- Performance Metrics: To achieve matching accuracy of at least 90% between LSP's delivery jobs and jobs from the DJE's job pool.
- Scalability: Able to handle large numbers of Pending Delivery Jobs & LSPs Jobs.
- User Interface: Display the list of recommended jobs for each LSP.

2. Literature Review

2.1. Introduction

The primary focus of the Outbound Job Recommender (OJR) Algorithm is to match LSPs with suitable jobs from the DJE job pool that aligns with their current delivery assignments. In line with this objective, this literature review aims to examine existing research on Delivery Job Matching (DJM), with a specific focus on investigating the utilization of technology and algorithms to optimize this process.

In this paper, DJM is defined as the allocation of appropriate jobs to suitable drivers or delivery personnel. DJM plays a pivotal role in logistics and supply chain management as it ensures efficient and timely deliveries while optimizing resource utilization. Consequently, the implementation of an effective DJM system can significantly enhance the delivery routing of LSPs, leading to improved operational efficiency.

Over the years, numerous algorithms have been developed to address the challenges associated with DJM. This review seeks to provide a comprehensive overview of the current state of knowledge in the field. By conducting a critical analysis of existing literature, the review will not only summarize the existing research but also identify gaps that warrant further investigation. By doing so, it contributes to the advancement of DJM research and the development of more effective matching systems.

2.2. Related Work

In [12], Gao discussed a scheduling algorithm that considers the constraint of a time window to choose an optimal taxi to serve the passenger in the context of ridesharing. However, this algorithm does not work when multiple constraints of a job need to be considered.

In [13], Chen presented a novel Stable Job Assignment Algorithm that achieves fair and non-wasteful job assignment results. In [14], Wang proposed the Personalized Taxi Matching Algorithm based on Top-K in the context of ridesharing, which matches taxis with requests. Later in [15], Lv also presented a novel Two-sided Stable Matching Algorithm, taking into account both workers' and requesters' preferences. However, since all three algorithms aim to recommend each job to multiple users, they may suffer from high computational complexity and limited scalability as the number of jobs and users increases.

In [16], Yan showcased a driver recommendation algorithm based on the kNN and decision tree algorithm, which dispatches orders based on the driver's performance on similar historical orders. This algorithm allows orders to be assigned to the most suitable drivers. However, it is not feasible when records of the driver's performance on historical orders are not available, and it also suffers from high computational complexity and limited scalability as the number of jobs and users increases.

In [17], Wang introduced a Deep Reinforcement Learning-based Order Recommendation (DRLOR) framework that enables efficient order recommendation in food delivery systems with a rider-centered assignment manner. However, since this work primarily focuses on the driver's behaviors leading to them taking up an order, it could potentially result in orders not being assigned to the most suitable driver for the job.

2.3. Key Findings

All the work mentioned above either considers only a single constraint, suffers from high computational complexity and limited scalability as the number of jobs and users increases, focuses on driver’s behaviors, which leads to jobs being matched to less suitable candidates, or requires the availability of historical data.

None of these approaches are suitable or feasible in the context of the DJE as the DJE operates in the logistic industry, where jobs or orders are parcels that need to be matched with a suitable LSP. Parcels in the logistic industry often consist of multiple features that must be considered as constraints to find the most suitable LSP for the job. For example, a parcel could have features such as "Frozen," "<60 cm (H+L+W) & max 1kg." Since this is a frozen food product parcel, it should only be assigned to LSPs with refrigerated vehicles. Hence, the OJR algorithm should be able to handle multiple constraints.

Secondly, the OJR algorithm should be less computationally intensive and scalable as the DJE deals with jobs at a company level, which means a larger number of jobs need to be considered.

Thirdly, the OJR algorithm should not only focus on the driver’s behaviors, which in the DJE’s context is the LSP’s behaviors. As the primary objective of the OJR is to match jobs to the most suitable LSPs, operating under the assumption that the higher the suitability of the job to the LSP, the greater the likelihood of the LSP accepting the job.

Lastly, the algorithm should be designed to match jobs with suitable LSPs without relying on the availability of historical data. Given the new concept of the DJE, it is assumed that historical data is not accessible in this context.

In accordance with the aforementioned requirements, this paper presents a pioneering approach in the logistic industry utilizing the graph-based algorithm, Spectral Clustering [18]. The Spectral Clustering algorithm satisfies all the specified requirements: it can accommodate multiple constraints, is computationally efficient and scalable, and can be readily customized to prioritize job matching based on the similarity between public jobs in the DJE job pool and the private jobs of the LSP, without relying on historical data.

3. Methodology

This section presents a novel methodology, addressing the limitations of existing literature in the field of Delivery Job Matching (DJM). The proposed approach, named the Outbound Job Recommender (OJR) Algorithm, leverages the graph-based algorithm, Spectral Clustering, to achieve efficient and effective matching of suitable jobs from the DJE job pool with LSPs. The OJR algorithm overcomes the deficiencies of previous approaches by considering multiple constraints, ensuring computational efficiency and scalability, focusing on the suitability of jobs for LSPs rather than LSP's behaviors, and operating without the need for historical data. By incorporating Spectral Clustering into the DJM process, the OJR algorithm introduces a novel methodology that addresses the unique requirements of the DJE, offering a promising solution for enhancing job allocation and resource optimization in the logistics and supply chain management industry. In the subsequent subsections, a comprehensive description of the OJR algorithm will be presented. This description will encompass the assumptions made, the overall flow of the algorithm, and detailed explanations of each key component within the flow.

3.1. Assumptions

To fully understand the decisions and choices made in the design process of the OJR algorithm, it is necessary to first understand the underlying assumptions that led to those choices. These assumptions serve as the foundation for the design and implementation of the OJR algorithm. It is essential to acknowledge and evaluate these assumptions to ensure the algorithm's reliability and effectiveness in practical scenarios. The following assumptions were made:

1) Availability of Real-Time Job Data: The OJR algorithm assumes the availability of real-time job data from the DJE. It assumes that the DJE provides accurate and up-to-date information regarding job details, including pickup and delivery locations, job features (e.g., frozen, capacity), and urgency levels. The algorithm relies on this assumption to ensure the relevance and timeliness of job recommendations provided to LSPs.

2) Accuracy of Job Features: The OJR algorithm assumes that all job features provided in the dataset, such as pickup/delivery locations, urgency, frozen status, and capacity, are accurate and reliable. Any inaccuracies or missing information may affect the quality of job matching and the algorithm's performance.

3) Accuracy of Postal Code to Coordinate Conversion: The algorithm assumes that the postal code to coordinate conversion function retrieves accurate latitude and longitude values based on the provided postal codes. Inaccurate conversions may lead to incorrect distance calculations and subsequently affect the job matching results.

4) Effectiveness of Gaussian Similarity Calculation: The OJR algorithm assumes that the Gaussian similarity calculation accurately captures the similarity between jobs based on their distances and urgency types. It assumes that the chosen values for sigma (σ) effectively produce meaningful similarity scores.

5) Effectiveness of Composite Adjacency Matrix: The algorithm assumes that combining the frozen, capacity, and distance adjacency matrices into a composite adjacency matrix provides a balanced representation of the job relationships. It assumes that averaging these matrices appropriately captures the different constraints and similarities between jobs, leading to effective spectral clustering.

6) Stable Company and Job Characteristics: The OJR algorithm assumes that the characteristics of companies and jobs remain stable during the matching process. It assumes that the company's capabilities, such as equipment and resources, and the job's requirements, such as frozen items and capacity, do not significantly change while the algorithm is running. This assumption allows for efficient matching based on initial data but may not account for dynamic changes in company or job characteristics during the allocation process.

7) Availability of Tailored Vehicles or Equipment: This assumption implies that LSPs maintain a fleet of vehicles or equipment tailored to their existing private jobs. For example, if an LSP has a private job involving parcels with features such as "Frozen: Yes" and "Capacity: 14Ft Lorry," it is assumed that the LSP has a refrigerated vehicle and a 14ft lorry readily available to handle similar jobs allocated from the DJE job pool. This assumption ensures that the recommended jobs align with the LSPs' capabilities and avoids assigning tasks that cannot be executed due to a lack of appropriate equipment. By assuming the availability of tailored vehicles or equipment, the OJR algorithm can prioritize job recommendations that match the specific requirements and capabilities of the LSPs.

8) Preferences of LSPs: This assumption implies that LSPs' preferences for certain jobs can be determined based on their existing private jobs. For example, if an LSP has a lot of private jobs involving parcels with the feature "Frozen: Yes," this assumption assumes that the LSP prefers jobs involving parcels with the same feature and will be willing to handle similar jobs allocated from the DJE job pool.

9) Job Relevance Assumption: It is assumed that the job relevance criteria used in the OJR algorithm accurately capture the capabilities, preferences, and requirements of both the LSPs and the jobs from the DJE job pool. The assumption is that the specified job features, such as urgency, capacity, frozen/non-frozen status, and geographic location, appropriately reflect the needs and constraints of the LSPs and enable effective job matching. It is assumed that these job features adequately capture the essential characteristics for job allocation.

10) It is assumed that the OJR algorithm will run every few hours to recommend and match the jobs in the DJE job pool to suitable LSPs. It is important to note that the LSPs will then have the same few hours' time window to accept the job offers before the OJR algorithm runs again to match the unaccepted jobs and new jobs in the DJE job pool to new suitable LSPs.

3.2. Outbound Job Recommender (OJR) Algorithm

Note that in future real-world scenarios, the number of public jobs and private jobs considered for deriving suitable matches in the DJE may be very large. Therefore, the OJR algorithm adopts an approach of matching each public job to the most suitable LSP instead of multiple LSPs, as done in previous literature. This approach allows for a reduction in computational intensity while ensuring the most accurate matching. The OJR Algorithm comprises key components, namely data collection and preprocessing, adjacency matrices construction, and spectral clustering. The following pseudo code provides an overview of the overall flow of the OJR algorithm before delving into each key component in detail.

3.2.1. *Overall Flow (Pseudo Code)*

Input: DJE job pool jobs data (public jobs) and each LSP's existing jobs data (private jobs)

Output: Returns a list of suitable public jobs for each LSPs

```
# Retrieve latitude and longitude using postal code
for public jobs V in V_all:
    call function getcoordinates() using V postal code, store results
for private jobs J in J_all:
    call function getcoordinates() using J postal code, store results

# Create frozen adjacency matrices
Initiate a matrix of zeros with number of rows and columns equals to total data
for row in range(len(frozen adjacency matrix)):
    for column in range(len(frozen adjacency matrix)):
        if row and column is the same job:
            continue to the next iteration
        else:
            Add an edge between rows and columns of private jobs with the same company id
            Add an edge between rows and columns,
            of all public and private job pairs that have the same frozen feature

# Create capacity adjacency matrices
Initiate a matrix of zeros with number of rows and columns equals to total data
for row in range(len(capacity adjacency matrix)):
    for column in range(len(capacity adjacency matrix)):
        if row and column is the same job:
            continue to the next iteration
        else:
            Add an edge between rows and columns of private jobs with the same company id
            Add an edge between rows and columns,
            of all public and private job pairs that have the same frozen feature
```

```

# Create distance adjacency matrices
Initiate a matrix of zeros with number of rows and columns equals to total data
for row in range(len(distance adjacency matrix)):
    for column in range(len(distance adjacency matrix)):
        if row and column is the same job:
            continue to the next iteration
        else:
            Calculate the distances of all possible routes,
            between public and private job pairs at the row and column,
            using haversine_distance() function
            Calculate gaussian_similarity() using the shortest route distance if the public job is urgent,
            else using longest route distance if the public job is not urgent
            add the gaussian similarity score as the edge value between the row and column

# Create composite adjacency matrix
Add an appropriate weight to each of the frozen, capacity, and distance adjacency matrices
Sum up the frozen, capacity, distance adjacency matrices and averaging these matrices into a single composite matrix

# Perform Spectral Clustering
Pass the composite adjacency matrix into the Spectral Clustering Algorithm
```

3.2.2. *Data Collection & Preprocessing*

The Data Collection & Preprocessing phase involves gathering the necessary data, including the jobs in the DJE job pool (referred to as public jobs) and the existing jobs held by each LSP (referred to as private jobs). The primary source for obtaining these data is the DJE. However, it is important to note that the DJE is currently a conceptual framework, and as

a result, the actual data required for the study are not yet available. Therefore, to facilitate the development of the algorithm, mock data based on real logistics industry information has been generated and utilized in this research.

Upon acquiring the necessary data, the OJR algorithm will initiate the process of data pruning. This step involves eliminating unnecessary columns or features from the data that are not crucial to the matching process, such as the service requestor's name and the price of the job. It will retain only the columns and features essential for the matching process, including Company Id, Order Id, Pickup Postal Code, Delivery Postal Code, Urgency, Capacity, and Frozen.

Next, the OJR will perform data cleansing to correct or remove errors in the dataset. For example, it will check the length of the postal codes in the dataset to ensure that they have a length of exactly 6. If there are jobs with postal codes of a length less than 6, it is often due to the removal of zeros at the front of the postal code. In such cases, the OJR will add zeros at the front of the postal code to correct the error.

Finally, the OJR will prepare the data by converting it to the correct format required for the algorithm to perform the job matching. For instance, this includes converting the postal codes to coordinates (latitude and longitude) format using available APIs such as OneMap's (Figure 3 shows a function making use of OneMap's API to convert postal codes to latitude and longitude) and storing the data in a data frame format.

```
14  # Function to retrieve latitude, longitude using postal code
15  def getcoordinates(address):
16      req = requests.get('https://developers.onemap.sg/commonapi/search?searchVal='+address+'&returnGeom=Y&getAddrDetails=Y&pageNum=1')
17      resultsdict = eval(req.text)
18      if len(resultsdict['results'])>0:
19          return resultsdict['results'][0]['LATITUDE'], resultsdict['results'][0]['LONGITUDE']
20      else:
21          pass
```

Fig 3. Function to convert postal codes to coordinates using OneMap's API

3.2.3. *Adjacency Matrices Construction*

In traditional Spectral Clustering algorithms, the adjacency matrix needs to be constructed based on the similarity between data points. This step

involves calculating pairwise similarities or distances between data points and determining the strength of the connections in the graph. The construction of the adjacency matrix plays a crucial role in capturing the underlying structure and relationships among data points, enabling Spectral Clustering to effectively identify clusters in the data.

Therefore, in the case of OJR, which is a Spectral Clustering-based algorithm, the construction of the adjacency matrix also plays a vital role. However, OJR adopts a slightly different approach. It first constructs multiple adjacency matrices, each considering the similarities between different features of the job data points, such as "Frozen," "Capacity," and "Urgency." The three adjacency matrices are then combined into a composite matrix by calculating the average of the three matrices. Prior to the averaging process, each adjacency matrix is multiplied by a corresponding weight based on its respective importance. This process aims to capture the diverse constraints and similarities between jobs, leading to effective spectral clustering. The following will provide a detailed description of how these adjacency matrices are constructed.

The Frozen Adjacency Matrix, denoted as F, represents an undirected graph with a vertex set $V = \{v_1, v_2, \cdots, v_n\}$. The Frozen Adjacency Matrix is a binary adjacency matrix, meaning that each edge between two vertices v_i and v_j is assigned a weight w_{ij}, which can only take the values 0 or 1. If $w_{ij} = 0$, it indicates that the vertices v_i and v_j are not connected by an edge. Conversely, if $w_{ij} = 1$, it indicates that the vertices v_i and v_j are connected by an edge. It's important to note that since F is an undirected graph, the weights must satisfy the condition $w_{ij} = w_{ji}$ to ensure symmetry in the adjacency matrix. The OJR construct the Frozen Adjacency Matrix by calling the create_frozen_adjacency_matrix() function, which takes in three parameters: `company_datapoints` representing the data points for the LSP's jobs, `job_datapoints` representing the data points for the DJE job pool's jobs, `features_matrix` the matrix used to store the common features between jobs. The function begins by calculating the number of total data points `datapoints_num`, which is the sum of the length of `company_datapoints` and `job_datapoints`. It then concatenates the

company and job data points into a single dataset `alljobs_datapoints`. A frozen adjacency matrix is created with dimensions `(datapoints_num, datapoints_num)` using NumPy. The matrix is initially filled with zeros. Next, the function iterates through the frozen adjacency matrix using nested loops. If the indices of both datapoints are the same `(i==j)`, it continues to the next iteration. Else for each pair of indices `(i, j)`, it checks if the indices correspond to company jobs or job trading jobs. If both indices correspond to company jobs, it checks if the company IDs of the two jobs are the same. If they are, the frozen adjacency matrix value at `(i, j)` is set to 1.0, indicating a connection. Additionally, the corresponding entry in the `features_matrix` is updated to include the company ID as a feature. If one index corresponds to an LSP/company job and the other corresponds to a DJE job pool's job, it checks if both jobs have the "Frozen" feature set to "Yes". If they do, the frozen adjacency matrix value at `(i, j)` is set to 1.0, indicating a connection as well. The `features_matrix` is also updated to include the feature "Frozen" for this pair of jobs. Finally, the function returns the frozen adjacency matrix. The frozen adjacency matrix represents the relationship between jobs based on the "Frozen" feature.

Like the Frozen Adjacency Matrix, the Capacity Adjacency Matrix, denoted as C, represents an undirected graph with a vertex set $V = \{v_1, v_2, \cdots, v_n\}$. The Capacity Adjacency Matrix is a binary adjacency matrix, meaning that each edge between two vertices v_i and v_j is assigned a weight w_{ij}, which can only take the values 0 or 1. If $w_{ij} = 0$, it indicates that the vertices v_i and v_j are not connected by an edge. Conversely, if $w_{ij} = 1$, it indicates that the vertices v_i and v_j are connected by an edge. It's important to note that since C is an undirected graph, the weights must satisfy the condition $w_{ij} = w_{ji}$ to ensure symmetry in the adjacency matrix. The OJR construct the Capacity Adjacency Matrix by calling the create_capacity_adjacency_matrix() function, which takes in three parameters: `company_datapoints` representing the data points for the LSP's jobs, `job_datapoints` representing the data points for the DJE job pool's jobs, `features_matrix` the matrix used to store the common features between jobs. The function begins by calculating the number of total data points `datapoints_num`, which is the sum of the length of

`company_datapoints` and `job_datapoints`. It then concatenates the company and job data points into a single dataset `alljobs_datapoints`. A capacity adjacency matrix is created with dimensions `(datapoints_num, datapoints_num)` using NumPy. The matrix is initially filled with zeros. Next, the function iterates through the capacity adjacency matrix using nested loops. If the indices of both datapoints are the same `(i==j)`, it continues to the next iteration. Else for each pair of indices `(i, j)`, it checks if the indices correspond to company jobs or job trading jobs. If both indices correspond to company jobs, it checks if the company IDs of the two jobs are the same. If they are, the capacity adjacency matrix value at `(i, j)` is set to 1.0, indicating a connection. Additionally, the corresponding entry in the `features_matrix` is updated to include the company ID as a feature. If one index corresponds to an LSP/company job and the other corresponds to a DJE job pool's job, it checks if both jobs have the same capacities. If they do, the capacity adjacency matrix value at `(i, j)` is set to 1.0, indicating a connection as well. The `features_matrix` is also updated to include the feature "Capacity" for this pair of jobs. Finally, the function returns the capacity adjacency matrix. The capacity adjacency matrix represents the relationship between jobs based on the "Capacity" feature.

The Distance Adjacency Matrix, denoted as D, represents an undirected graph with a vertex set $V = \{v_1, v_2, \ldots, v_n\}$. The Distance Adjacency Matrix is a weighted adjacency matrix, meaning that each edge between two vertices v_i and v_j is assigned a weight w_{ij}, which can only take non-negative values $w_{ij} \geq 0$. If $w_{ij} = 0$, it indicates that the vertices v_i and v_j are not connected by an edge. It's important to note that since D is an undirected graph, the weights must satisfy the condition $w_{ij} = w_{ji}$ to ensure symmetry in the adjacency matrix. The OJR construct the Distance Adjacency Matrix by calling the create_distance_adjacency_matrix() function, which takes in two parameters: `company_datapoints` representing the data points for the LSP's jobs, `job_datapoints` representing the data points for the DJE job pool's jobs. The function begins by calculating the number of total data points `datapoints_num`, which is the sum of the length of `company_datapoints` and `job_datapoints`. It then concatenates the company and job data points into

a single dataset `alljobs_datapoints`. A distance adjacency matrix is created with dimensions `(datapoints_num, datapoints_num)` using NumPy. The matrix is initially filled with zeros. Next, the function iterates through the distance adjacency matrix using nested loops. If the indices of both datapoints are the same `(i==j)`, it continues to the next iteration. Else for each pair of indices `(i, j)`, it checks if the indices correspond to company jobs or job trading jobs. If both indices correspond to company jobs, it checks if the company IDs of the two jobs are the same. If they are, the distance adjacency matrix value at `(i, j)` is set to 1.0, indicating a connection. If one index corresponds to an LSP/company job and the other corresponds to a DJE job pool's job, it calculates the distances between the pickup and delivery locations of the two jobs. It considers all possible route permutations between the two jobs and calculates the distances for each permutation using the haversine_distance() helper function. The haversine_distance() function calculates the haversine distance between two sets of latitude and longitude coordinates taking in parameters of lat1 (float): Latitude of the first job, lon1 (float): Longitude of the first job, lat2 (float): Latitude of the second job, lon2 (float): Longitude of the second job. The OJR then considers the urgency type of the job from the DJE job pool, If the job is "Urgent," it calculates the shortest route distance among the permutations and uses it to calculate the Gaussian similarity using the gaussian_similarity() helper function. If the job is "Not Urgent," it calculates the longest route distance among the permutations and uses it to calculate the complementary Gaussian similarity. The gaussian_similarity() function calculates the Gaussian similarity between two jobs based on the distance, urgency type, and sigma value taking in parameters of distance (float): Distance between two jobs, urgency_type (string): Type of urgency ("Urgent" or "Not Urgent"), sigma (float): Sigma value for Gaussian similarity calculation. The distance adjacency matrix value at `(i, j)` is then set to the derived similarity value, the closer the similarity value to 1 indicates a higher similarity between the two jobs. Finally, the function returns the distance adjacency matrix. The distance adjacency matrix represents the relationship between jobs based on the "Urgency" feature.

After obtaining the frozen, capacity, and distance adjacency matrices, the

OJR algorithm proceeds to calculate the composite adjacency matrix, which can be expressed using the following formula:

$$CAM = \frac{w_1[F] + w_2[C] + w_3[D]}{3}$$

Each adjacency matrix (frozen adjacency matrix F, capacity adjacency matrix C, distance adjacency matrix D) is first multiplied by a weight based on its respective importance, denoted as w_1, w_2, and w_3, respectively. For instance, if it is crucial to assign public jobs with the "Frozen" feature to LSPs with similar "Frozen" feature jobs, a higher value should be assigned to the corresponding weight of the frozen adjacency matrix (e.g., $w_1 = 2$, $w_2 = 1$, $w_3 = 1$). The element-wise mean of the three matrices is then calculated by summing them and dividing the result by three. This process effectively merges the three matrices into a single composite adjacency matrix.

3.2.4. Spectral Clustering

OJR then performs Spectral Clustering using the composite adjacency matrix with the following steps:

1) Compute the graph Laplacian Matrix, L, which is defined as $L = D - C$, where C represents the derived Composite Adjacency Matrix and D is a Diagonal Matrix where D_{ii} is the sum of the i-th row (or column) of C. The Laplacian Matrix captures the structural information of the data and is also essential for Spectral Clustering.

2) Compute the eigenvectors and eigenvalues of the Laplacian Matrix L. Let $\lambda 1, \lambda 2, \ldots, \lambda \mathrm{N}$ be the eigenvalues in ascending order and $\mathrm{v}1, \mathrm{v}2, \ldots, \mathrm{vN}$ be the corresponding eigenvectors.

3) Select k eigenvectors corresponding to the k smallest eigenvalues (excluding the smallest eigenvalue, which corresponds to the constant vector). Arrange these eigenvectors as columns in a matrix V.

4) Normalize the rows of matrix V, forming a new matrix U.

5) Apply a clustering algorithm, such as k-means, to the rows of U in order to assign data points to clusters. In this context, the number of clusters, denoted as k, will be defined as the number of LSPs/companies, with each cluster containing jobs matched for one company.

6) The resulting clusters represent the final output of the Spectral Clustering algorithm, with each cluster being assigned to its respective LSPs/companies.

The completion of the Spectral Clustering step marks the end of the OJR algorithm, where the algorithm successfully generates relevant job clusters for each LSP. Each cluster is intended to be recommended to exactly one LSP. To analyze the results of the OJR, the python library NetworkX can be utilized to plot graphs that illustrate the relationships between the features of the jobs within the generated clusters, as depicted in Figure 4. These graphs are constructed based on the features matrix, which is filled during the construction of the adjacency matrices. Figure 5 provides an example of a zoomed-in view showcasing the feature relationships between jobs in a particular cluster, specifically the cluster associated with LSP "Company Id" 4. The visualizations reveal that the jobs are correctly matched, as all the public jobs within a given cluster share common features with at least one private job in the same cluster.

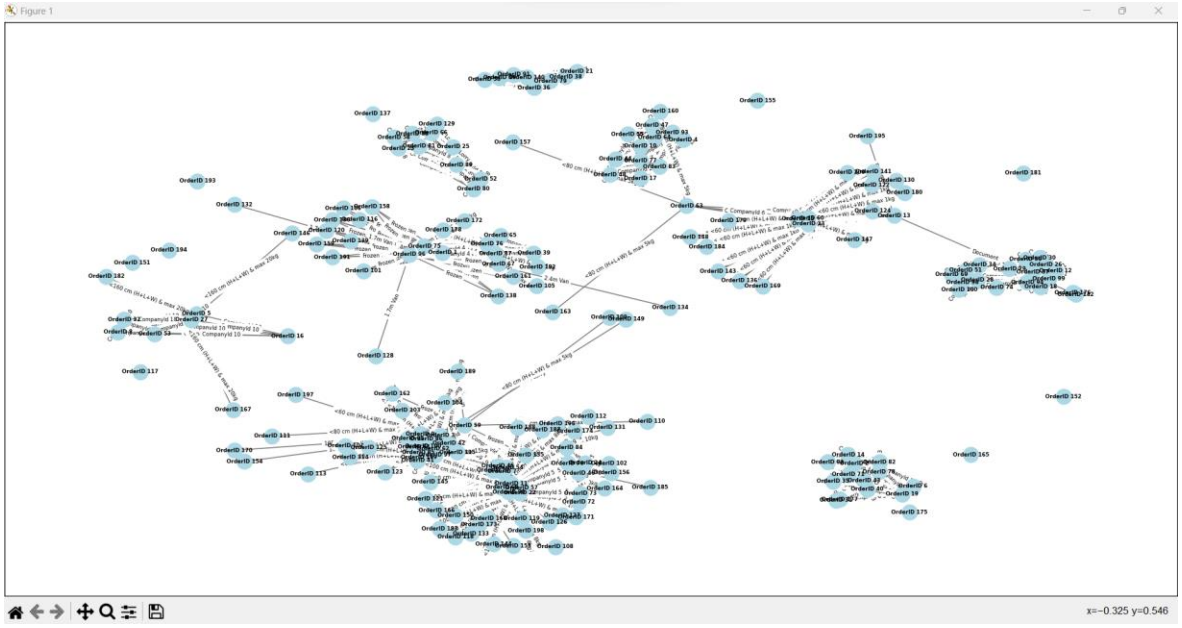

Fig 4. Features Relationship Graph (All Clusters)

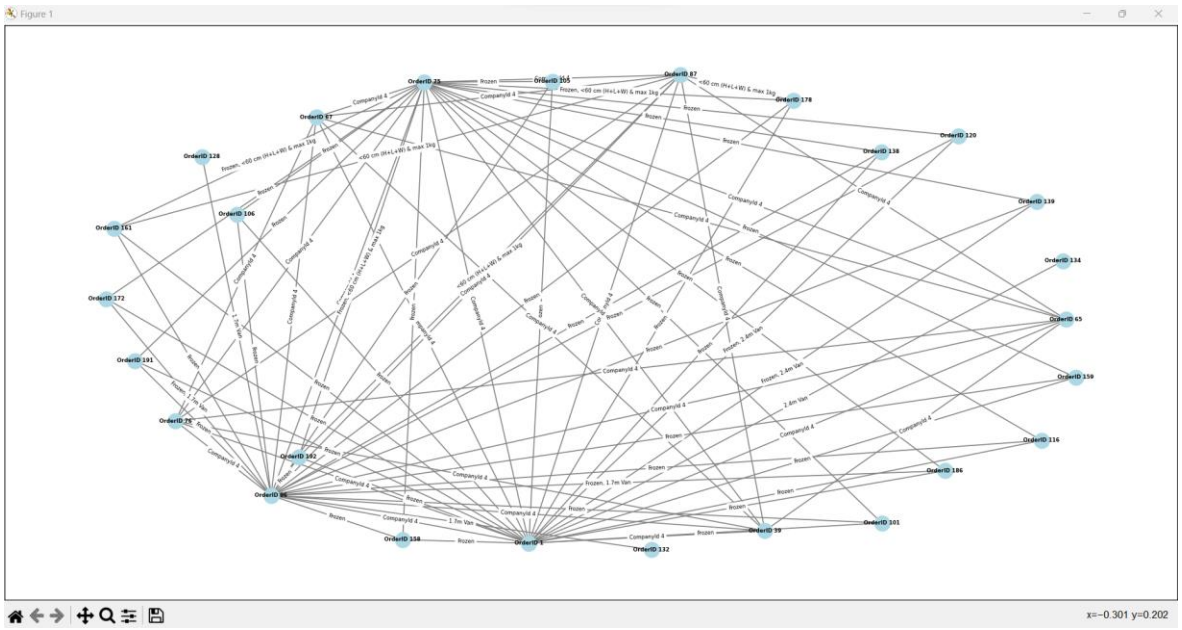

Fig 5. Features Relationship Graph (Company Id 4)

4. Results and Analysis

In this specific academic paper focusing on job matching and clustering between a pool of publicly available jobs and LSPs in the logistic industry, there is a lack of relevant research in this specific context. Therefore, to establish a benchmark for comparison, the paper will employ the k-means algorithm. The choice of the k-means algorithm is justified by its common usage as a clustering algorithm. By utilizing k-means as a benchmark, the proposed solution can be evaluated and analyzed in relation to a well-established and widely used clustering algorithm. This approach allows for a meaningful comparison between the proposed algorithm and an established method, shedding light on the performance and effectiveness of the new approach in the specific domain of job matching and clustering in the logistics industry.

4.1. Results

In this section, the findings of the research will be reported.

4.1.1. Clustering Entropy

After executing the proposed algorithm and plotting the clustering results of all public and private jobs on a map, as illustrated in Figure 6 and Figure 7 below, it is evident that the algorithm successfully maintains the initial private jobs of the LSPs. This outcome aligns with the expectations and requirements of the research, as LSPs have a vested interest in retaining their private jobs due to factors such as confidentiality.

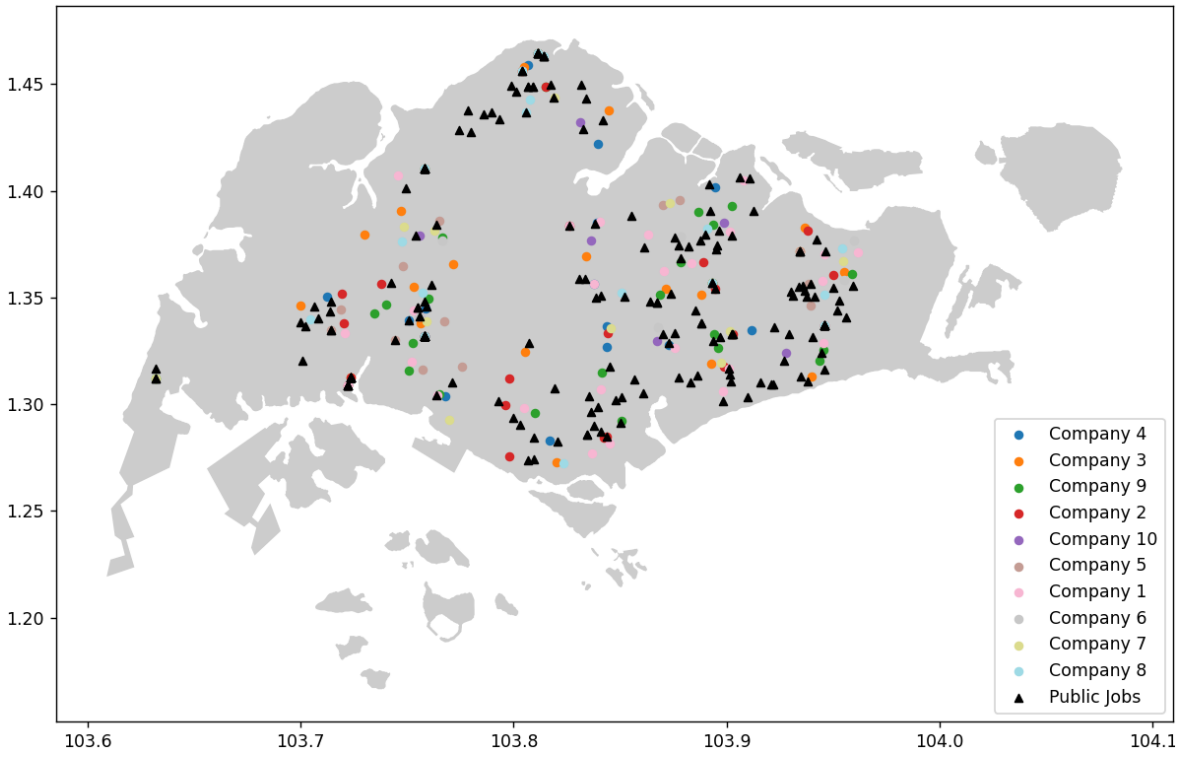

Fig 6. 100 Public & 100 Private Jobs Map Plot (Before OJR)

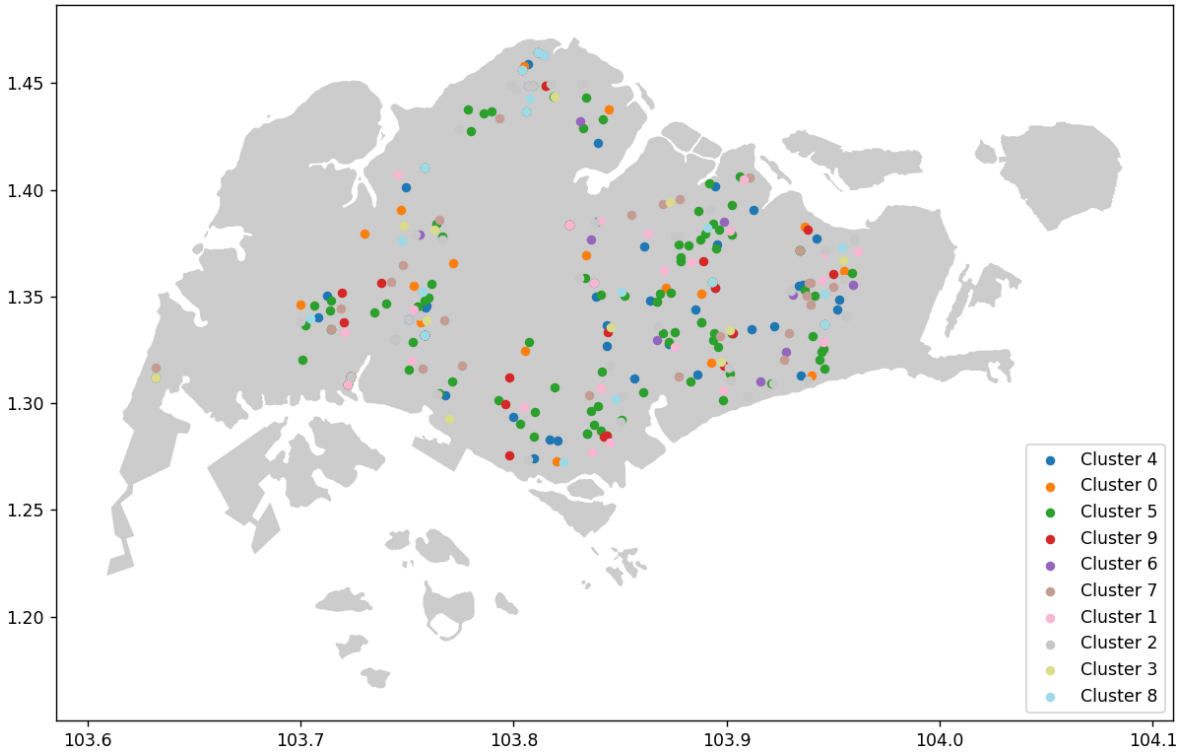

Fig 7. 100 Public & 100 Private Jobs Map Plot (After OJR)

At times, it may appear that the proposed algorithm assigns more jobs to a particular cluster, such as Cluster 5 (Company 9), as depicted in Figure 7. However, this occurrence is attributable to characteristics inherent in the dataset. For instance, a specific company may possess a larger number of jobs (private jobs) with features highly similar to those found in the DJE job pool (public jobs). This outcome is expected since the requirement is to match public jobs with the most suitable Local Service Provider (LSP) based on their private jobs.

To assess the quality of the clustering and compare the proposed algorithm to the k-means algorithm, the entropy metric can be employed. The entropy metric quantifies the clustering results' quality, with lower entropy values indicating a higher level of cluster purity and better separation of data points into distinct clusters. The entropy metric is defined by the formula:

$$-\sum_{j \in K} P(i_j) \log_2 P(i_j)$$

The above formula can be explained as follows: Let K be a set of unique cluster labels obtained after running an algorithm, where j represents a cluster label in the set K. To calculate the entropy of the cluster result, compute the probability of each cluster label in set K, multiply it by the binary logarithm of the probability, and repeat this process for every cluster label in set K. The resulting values are then summed. For example, if the cluster labels obtained are [1, 2, 2, 2, 2, 3], the entropy is calculated as $-\left(\left(\frac{1}{6}\log\frac{1}{6}\right)+\left(\frac{4}{6}\log\frac{4}{6}\right)+\left(\frac{1}{6}\log\frac{1}{6}\right)\right) = 1.0986122886681096.$

To visualize the entropy results obtained from running the proposed algorithm and the k-means algorithm, a line graph is plotted using data points derived from 100, 500, and 1000 public jobs, respectively.

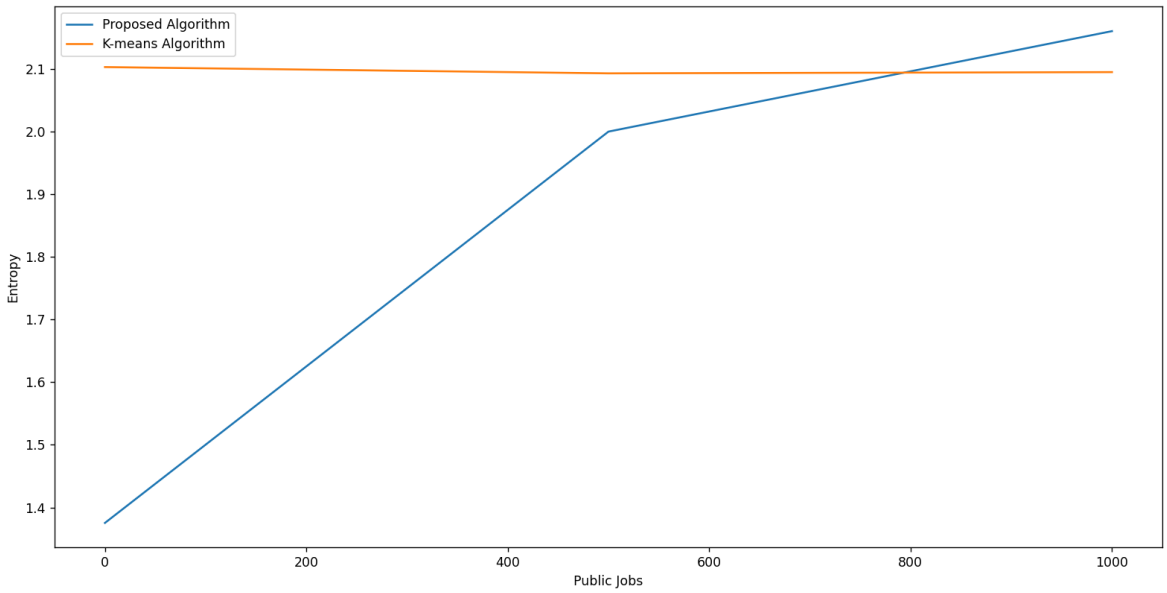

Fig 8. Entropy Comparison

The entropy values observed for k-means were 2.103033521165634 for 100 public jobs, 2.093035006031204 for 500 public jobs, and 2.0950133366487678 for 1000 public jobs. Meanwhile, the entropy values recorded for the proposed algorithm, OJR, were 1.375312701275081 for 100 public jobs, 2.00003085633245 for 500 public jobs, and 2.160420059674329 for 1000 public jobs.

When comparing the entropy values between OJR and k-means, it is evident that OJR consistently outperforms k-means in terms of cluster quality. The entropy values obtained with OJR are consistently lower than those obtained with k-means, indicating superior separation and higher cluster purity.

Additionally, a decreasing trend in the entropy values can be observed as the dataset size increases for OJR. This suggests that the algorithm benefits from larger datasets, as it achieves better-defined clusters with reduced uncertainty, which is highly suitable for the DJE context.

Overall, these results demonstrate the superiority of the proposed OJR algorithm over the traditional k-means algorithm in terms of cluster quality. The lower entropy values obtained with OJR indicate more accurate and well-separated clusters, making it a promising and suitable approach for

contexts such as DJE.

4.1.2. Run Time

Next, to assess the performance and scalability of OJR, the runtime of OJR will be compared to that of the k-means algorithm.

The following line graph illustrates the runtime results obtained from running the proposed algorithm and the k-means algorithm with 100, 500, and 1000 public jobs data points.

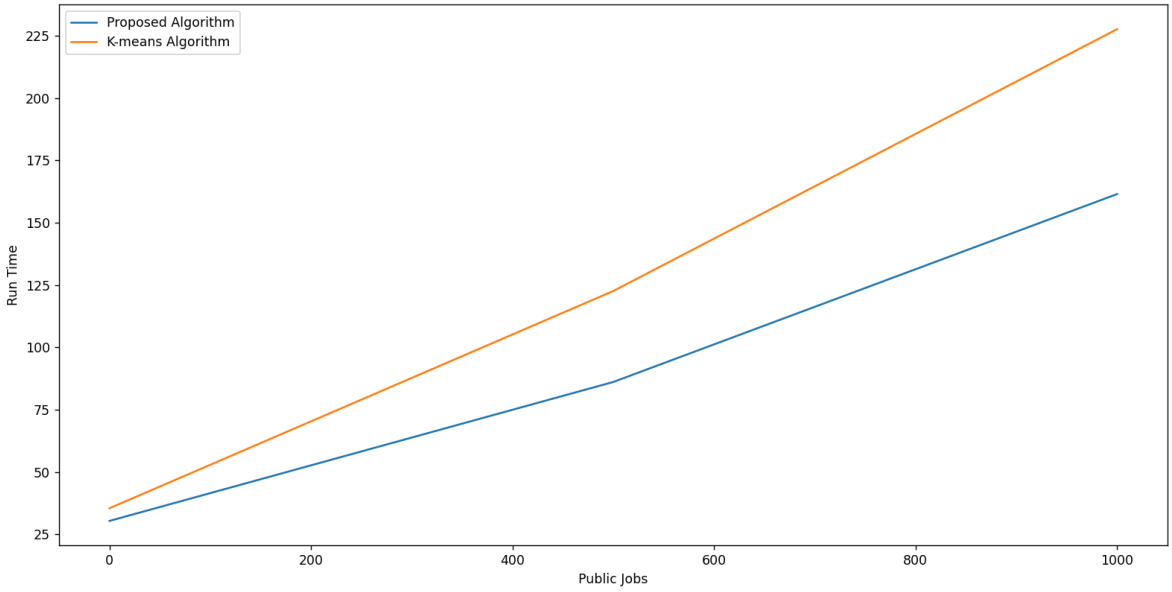

Fig 9. Run Time Comparison

The runtime values observed for k-means were 35.49083948135376 seconds for 100 public jobs, 122.65361475944519 seconds for 500 public jobs, and 227.6941328048706 seconds for 1000 public jobs. Conversely, the runtime values observed for OJR were 30.418132305145264 seconds for 100 public jobs, 86.15991139411926 seconds for 500 public jobs, and 161.5476655960083 seconds for 1000 public jobs.

Comparing the runtime results between OJR and the k-means algorithm, it is evident that OJR consistently demonstrates faster execution times across all dataset sizes. The runtime of OJR is consistently lower than that of the k-means algorithm, indicating improved computational efficiency.

Although, like the k-means algorithm, the runtime of OJR increases with the dataset size, it is noteworthy that OJR continues to outperform k-means in terms of runtime efficiency, even for larger datasets.

Therefore, the runtime analysis suggests that OJR provides a computationally efficient and scalable solution for the DJE context when compared to the traditional k-means algorithm, even when dealing with larger datasets. This feature makes OJR well-suited for effectively handling real-world data with thousands of data points, which is characteristic of the DJE domain. Additionally, the reduced runtime of OJR will prove highly advantageous in the DJE context, where prompt delivery is frequently required for time-sensitive delivery jobs.

4.2. Analysis

Based on the aforementioned findings, the subsequent section will present an analysis of the results.

4.2.1. Weaknesses of OJR

1) The algorithm occasionally assigns more jobs to certain clusters, which may result in uneven distribution and potential bias towards specific companies or clusters.

2) The entropy values for OJR are higher compared to k-means for larger datasets, indicating a decrease in cluster purity and separation as the dataset size increases.

4.2.2. Strengths of OJR

1) The algorithm successfully maintains the initial private jobs of the LSPs, aligning with the research expectations and requirements.

2) OJR consistently outperforms k-means in terms of cluster quality, as evidenced by lower entropy values. This indicates better separation and higher cluster purity with OJR.

3) OJR demonstrates faster runtime efficiency across all dataset sizes compared to the traditional k-means algorithm.

4.2.3. Discussion of the Effect and Impact

The findings suggest that the proposed OJR algorithm has a positive impact in the context of the DJE domain. By maintaining the initial private jobs of LSPs, the algorithm respects confidentiality and meets the requirements of the research. Additionally, the algorithm's ability to achieve better cluster quality and separation enhances the accuracy and effectiveness of job assignments, leading to improved service delivery in the DJE industry.

The faster runtime efficiency of OJR is a significant advantage, especially in a time-sensitive domain like DJE where prompt delivery is crucial. The reduced computational time allows for improved operational efficiency, contributing to better overall performance in managing delivery jobs.

4.2.4. Criticism

Despite the strengths mentioned above, there are areas for improvement and potential criticism. The uneven distribution of jobs to certain clusters, as observed in the data, raises concerns about potential bias and fairness in job assignments. It would be beneficial to investigate and address the factors that lead to such imbalances in order to ensure equal opportunities for all LSPs and prevent any preferential treatment.

The increasing entropy values for larger datasets with OJR suggest that there may be limitations to its scalability. It is important to explore strategies to mitigate this issue and maintain the quality of clustering as the dataset size grows.

Hence, while the proposed OJR algorithm demonstrates several strengths such as maintaining private jobs, superior cluster quality, and faster runtime efficiency, there are areas that require attention and further development.

5. Project Management

Throughout the project duration, I maintained a personal schedule to ensure the completion of weekly tasks, which collectively contributed to the overall project completion. Figure 10, the Gantt Chart, provides an overview of the project's stages and their respective accomplishments. This section will describe each stage in detail.

5.1. Research

During this phase, extensive research was conducted by reviewing existing literature and online resources. Research activities persisted throughout the project, as continuous knowledge improvement on the topic was necessary. Initially, research was required to identify a suitable research topic. Subsequently, further research was undertaken to enhance understanding of the chosen topic. During the planning and algorithm design phase, additional research was conducted to explore potential solutions for gaps identified in the existing literature. Research was also conducted to identify appropriate tools, implementation methodologies, and potential metrics for the analysis phase.

5.2. Planning & Algorithm Design

In this phase, a comprehensive plan was formulated for algorithm implementation. This involved making decisions regarding necessary assumptions and requirements. Planning was crucial to avoid impulsive actions and ensure the efficient utilization of time. It helped prevent unnecessary implementation or omission of essential components, which could have caused disruptions. The algorithm's design, including potential implementation approaches and solution considerations, was carefully determined during this stage.

5.3. Tool Selection

The tool selection stage involved making informed decisions on the specific tools to be used for algorithm development. This encompassed choosing programming languages, libraries, and other necessary resources for the implementation. As the project's focus was on software development for the algorithm, no budget allocation or physical tools were required.

5.4. Implementation

This phase constituted a significant portion of the project timeline, as it involved the actual implementation of the algorithm. The implementation was based on the accumulated knowledge from the research phase, the planned algorithm flow, the established assumptions, the defined requirements, and the selected tools. Consultations with the academic supervisor were also conducted to ensure alignment with project objectives and to identify potential areas for improvement.

5.5. Results Analysis

The final phase involved analyzing the outcomes of the algorithm implementation. A comparison was made between the results obtained from the implemented algorithm and the commonly used k-means algorithm. This comparison aimed to assess the performance of the developed algorithm and identify its strengths and weaknesses. The findings from this analysis will serve as valuable insights for future implementations, informing strategies for further enhancing the algorithm.

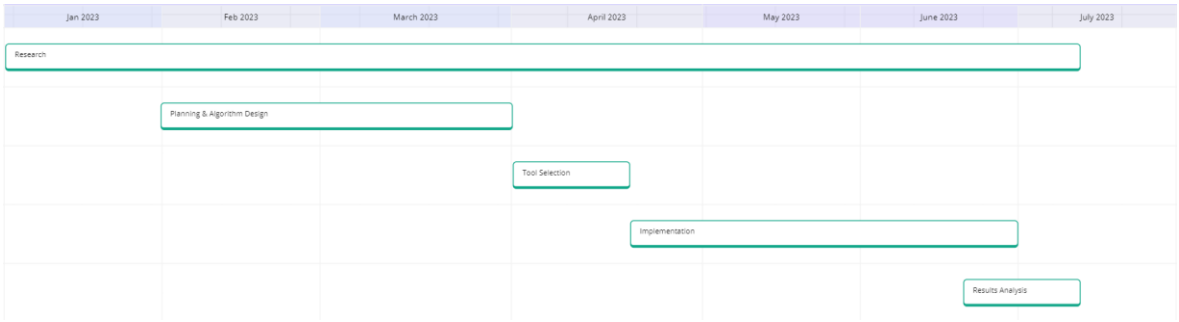

Fig 10. Gantt Chart

6. Conclusion

This research paper introduces the Outbound Job Recommender (OJR) algorithm, which aims to match Logistic Service Providers (LSPs) with suitable jobs from a shared job pool known as DJE. The algorithm demonstrates superior cluster quality compared to the traditional k-means algorithm, as evidenced by lower entropy values. Additionally, OJR exhibits faster runtime efficiency, providing a computationally efficient and scalable solution for the DJE context, even with larger datasets.

The OJR algorithm successfully tackles the challenge of efficiently and accurately matching Logistic Service Providers (LSPs) with suitable delivery jobs from the DJE job pool, all while fulfilling the specified requirements. It effectively handles multiple constraints, maintains computational efficiency and scalability, and prioritizes job matching based on the similarity between public jobs in the DJE job pool and the private jobs of the LSP, without relying on historical data. These characteristics make the OJR algorithm highly suitable for the DJE context, as it enhances delivery routing and operational efficiency by allowing LSPs to utilize idle delivery drivers from other LSPs or companies.

To further enhance the OJR algorithm, several directions for future work can be explored. Firstly, it is important to investigate and address any occasional uneven distribution of jobs to certain clusters, ensuring fair and unbiased job assignments. Secondly, strategies should be developed to improve scalability and reduce entropy values for larger datasets, enabling the algorithm to handle a higher number of jobs more efficiently. Additionally, considering dynamic factors such as real-time traffic conditions or weather can enhance the accuracy and adaptability of the job matching process. Pursuing these avenues for future work will contribute to continuous improvements and optimization of the OJR algorithm, resulting in enhanced job matching outcomes and further advancements in the DJE domain.

7. References

[1] “What is a Logistics Service Provider (LSP)?,” Penske Logistics, https://www.penskelogistics.com/insights/logistics-glossary/what-is-a-logistics-service-provider (accessed Jul. 8, 2023).

[2] Shipsy, “The world’s leading logistics software provider,” Shipsy, https://shipsy.io/blogs/7-common-challenges-faced-by-courier-service-providers/ (accessed Jul. 8, 2023).

[3] “Six key trends impacting global supply chains in 2022,” KPMG, https://kpmg.com/sg/en/home/insights/2022/03/six-key-trends-impacting-global-supply-chains-in-2022.html#:~:text=Driver%20shortages%2C%20logistics%20provider%20capacity,drivi (accessed Jul. 8, 2023).

[4] M. Curoe, “Common logistics problems and how to solve them,” Redwood Logistics, https://www.redwoodlogistics.com/common-logistics-problems-and-how-to-solve-them/ (accessed Jul. 8, 2023).

[5] A. Bio, “Top causes of late delivery and how to overcome them,” Upper Route Planner, https://www.upperinc.com/blog/causes-of-late-delivery/ (accessed Jul. 8, 2023).

[6] “10 common causes of delayed deliveries - shiptrack – track anything, anywhere!,” ShipTrack, https://shiptrackapp.com/blog/10-common-causes-of-delayed-deliveries/ (accessed Jul. 8, 2023).

[7] O. Blog, “Crowdsourcing becoming a solution in logistics, other industries,” Little Rock and Fayetteville AR Delivery Services, http://www.otlusa.biz/crowdsourcing-becoming-a-solution-in-logistics-other-industries/ (accessed Jul. 8, 2023).

[8] Farbod, “Multi apping: The art of driving for Multiple Delivery Apps & Services,” Moves, https://movesfinancial.com/blog/multi-apping-delivery-apps/ (accessed Jul. 8, 2023).

[9] B. Harvey, “The advantages of driving for multiple delivery companies,” Inshur, https://inshur.com/blog/the-advantages-of-driving-for-multiple-delivery-companies/ (accessed Jul. 8, 2023).

[10] “What’s wrong with route optimization? A gap between theory and Reality,” Kardinal, https://kardinal.ai/whats-wrong-with-route-optimization/ (accessed Jul. 8, 2023).

[11] “How much is inefficient delivery route planning costing you?,” Aptean.com,

https://www.aptean.com/en-US/insights/blog/inefficient-delivery-route-planning-cost (accessed Jul. 8, 2023).

[12] J. Gao, Y. Wang, H. Tang, Z. Yin, L. Ni and Y. Shen, "An Efficient Dynamic Ridesharing Algorithm," 2017 IEEE International Conference on Computer and Information Technology (CIT), Helsinki, Finland, 2017, pp. 320-325, doi: 10.1109/CIT.2017.33.

[13] Y. Chen and X. Yin, "Stable Job Assignment for Crowdsourcing," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, 2017, pp. 1-6, doi: 10.1109/GLOCOM.2017.8254454.

[14] L. Wang, F. Su, J. Dong and J. Li, "A Personalized Taxi Matching Algorithm for Ridesharing," 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC), Guiyang, China, 2018, pp. 65-70, doi: 10.1109/ICNIDC.2018.8525801.

[15] J. Lv, J. Hao, S. Yao and H. Tan, "A Two-Sided Stable Matching Method in Ridesharing," 2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems (CCIS), Chengdu, China, 2022, pp. 671-675, doi: 10.1109/CCIS57298.2022.10016360.

[16] H. Yan and H. Wang, "Design and Implementation of Driver Recommendation Algorithm in Enterprise Transportation," 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2021, pp. 1373-1378, doi: 10.1109/IAEAC50856.2021.9390823.

[17] X. Wang, L. Wang, C. Dong, H. Ren and K. Xing, "An Online Deep Reinforcement Learning-Based Order Recommendation Framework for Rider-Centered Food Delivery System," in IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 5, pp. 5640-5654, May 2023, doi: 10.1109/TITS.2023.3237580.

[18] On spectral clustering: Analysis and an algorithm - neurips, https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf (accessed Jul. 14, 2023).

8. Knowledge and Training Requirements

8.1. Applicable Knowledge from the Degree Programme

The prerequisite knowledge and skillsets from the degree programme that was necessary for the capstone projects are as follows:

No.	Module(s)	Knowledge(s) Applied
1	CSC1006: Mathematics 2	The knowledge gained from Mathematics 2 provided me with a solid foundation in the essential mathematical concepts pertaining to eigenvalues and eigenvectors. This foundational understanding played a crucial role in comprehending the intricacies of the Spectral Clustering algorithm. By possessing this fundamental knowledge, I was able to grasp the algorithm effortlessly, ultimately leading to significant facilitation of my research.
2	CSC1008: Data Structures and Algorithms	The study of Data Structures and Algorithms heightened my awareness of the computational efficiencies inherent in program design. It significantly enriched my knowledge regarding the development of efficient solutions and the critical analysis of existing algorithms to identify potential efficiency issues. This knowledge proved invaluable throughout the duration of my capstone project, as it enabled me to identify gaps in existing literature and conceive innovative approaches for addressing the research problem at hand.
4	CSC2011, CSC2012: Professional Software Development	This module has provided me with invaluable knowledge in software engineering principles, software architecture, design patterns, testing, and quality assurance. This knowledge proved indispensable during the design phase of my innovative solution, enabling me to adhere to engineering principles, best practices, and ultimately design an effective and efficient solution.
5	CSC3005: Data Analytics	Data Analytics has equipped me with the necessary skills to collect, clean, analyze, visualize, and interpret data using statistical techniques and machine learning algorithms. This knowledge has proven to be particularly valuable during the Data Collection & Preprocessing stage of my project. It has enabled

	me to effectively clean and analyze the data, ensuring its suitability for my specific use case.

8.2. Additional Knowledge, Skillsets, or Certifications Required

The following are the additional requirements and knowledge that were required for the capstone projects:

No.	Additional Requirement(s)	Knowledge(s) Applied
1	Logistic Industry Knowledge	Logistic industry knowledge plays a vital role in this project as it enables a comprehensive understanding of the current industry landscape and the challenges it presents. Moreover, this knowledge is invaluable during the ideation phase as it guides the decision-making process in selecting the most effective and feasible implementation approaches to address the target problem.

END OF REPORT