EasyChair Preprint

№ 10897

Clustering Techniques to Optimize Railway Daily Path Utilization for Non-Daily Trains

Karim Shahbaz, Mohit Agarwala, Samay P. Singh,
Satwik V. Ramisetty, Sayali R. Duragkar, Merajus Salekin,
Raja Gopalakrishnan, Narayan Rangaraj and Madhu N. Belur

September 13, 2023

World Conference on Transport Research - WCTR 2023 Montreal 17-21 July 2023

Clustering techniques to optimize railway daily path utilization for non-daily trains

Karim Shahbaz[a], Mohit Agarwala[a], Samay P. Singh[a], Satwik V. Ramisetty[a], Sayali R. Duragkar[a], Merajus Salekin[c], Raja Gopalakrishnan[c], Narayan Rangaraj[b], Madhu N. Belur[a]

[a]*Department of Electrical Engineering, Indian Institute of Technology Bombay, India*

[b]*Industrial Engineering and Operation Research, Indian Institute of Technology Bombay, India*

[c]*Centre for Railway Information Systems, New Delhi, India*

Abstract

A typical rail-network has a combination of daily trains and non-daily trains that have a weekly-pattern. Being non-daily, such trains run sporadically across the week and thus creates the problem of inefficient timetabling. Further, for large rail-networks, the timetabling is often done decentrally zone-wise without explicitly ensuring that groups of non-daily trains have similar running times on the bottleneck sections. In this paper, we use a notion of 'dailyzing' (making a daily path of non-daily trains) by performing a modulo 24 hours operation and then using Hierarchical Agglomerative Clustering (amongst other techniques) to group the trains. These clusters of trains share the same railway resources almost simultaneously but on different days of the week. Thus the scheduling of one representative train of the cluster as a 'daily train' would automatically schedule non-daily ones in that group. Hence, the daily path for non-daily trains provides a systematic and more efficient way of timetabling. The clustering/grouping of trains can also help find an empty slot for a new train scheduling/addition or help in point towards resource under-utilization.

Keywords: Train timetabling; Clustering techniques in train-timetabling; Hierarchical clustering

1. Introduction

Indian Railways operates about 13,150 passenger trains daily [6]; many of them run every day while others run non-daily, but repeat with a weekly-pattern. The non-daily trains run once, twice or multiple times per week. Daily trains' timetabling is fairly efficient and compact, but non-daily trains timetabling is not as much; this is due to the non-daily trains being sparsely

E-mail address: karimshahee@ee.iitb.ac.in

spread throughout the week and having very heterogeneous running patterns. The individual non-daily trains could have running variations with some others on common sections of their route. This variation leads to the difficulty of timetabling them efficiently and systematically to achieve better resource-utilization of key-bottleneck sections in the route.

The objective we pursue in this paper is to analyze the data and use the data to group non-daily trains in the Indian Railway Golden Quadrilateral and Diagonals (GQD) data to thus detect under-utilized track/platform resources and speed up timetabling efficiently. The need to extract this information from the data arises since the timetabling process for large rail-networks is done decentrally and zone-wise and hence the grouping process of non-daily trains is not utilized sufficiently during the timetabling process.

Once the grouping information is extracted from the data, in addition to more systematic use of this information in the timetabling process, the grouping would also aid in identifying regions where more trains can be deployed, and track-resources, especially in the bottleneck sections, could then be used more efficiently.

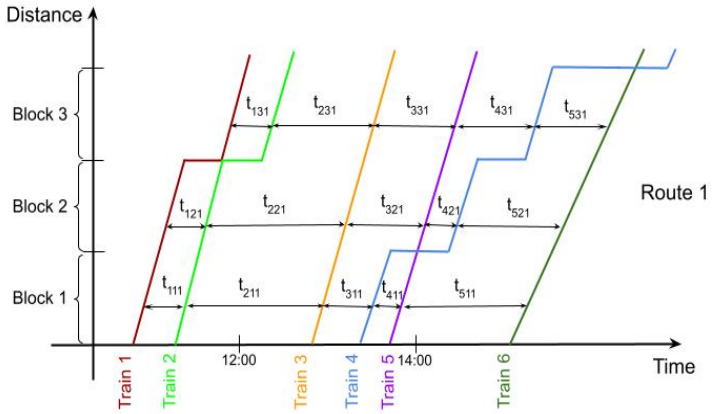

Figure 1: Distance vs time chart: each trajectory is a 'path' (see Definition 1.1)

Figure 1 consists of various trajectories of train running: distance versus time. At any station or block-section, i.e. the part of the rail-track between two stations, the horizontal distance between two consecutive train 'paths' (see Definition 1.1 below) is the time-gap between the trains: this needs to be higher than a minimum safety time value.

Definition 1.1. *In the distance vs time chart (see Figure 1, for example), a 'path' (in usual railway parlance) is defined as a trajectory/curve followed by a train in this chart.*

The notion of 'compaction' plays a central role in any timetabling process. This is included as a definition.

Definition 1.2. *Compaction is a phenomenon of achieving closeness in the horizontal axis between the paths in the distance vs time chart. Thus, on an average, steeper, i.e. closer to vertical, paths would lead to more number of paths within the same time-window, and thus more compaction.*

Achieving more compaction within a timetable is the objective in any timetabling process: better compaction results in better resource utilization and also a wider time-window to perform (preventive) planned maintenance activities. Steeper paths means higher average running times and thus higher Quality of Service to the end-user/customer, and also quicker turn-around for the train rakes.

In the context of this paper, we focus on grouping those trains that use almost the same daily 'path' (in the sense of Definition 1.1) but on different days of the week. Thus the members in a group need to *complement* each other with respect to the week-days. Since we need this condition repeatedly throughout this paper, we define it below. This is relevant since the clustering techniques would result in clusters that would later need to be imposed with the complementing condition.

Definition 1.3. Complementing condition: *a cluster is said to be satisfying the complementing condition if every member of the cluster utilizes almost the same resources and at almost the same times, but on different days of the week, i.e. they complement each other in terms of days of the week. We say the complementing condition is not satisfied by a group if the group consists of some trains running on the same days at almost the same time, on almost the same block-section.*

Of course, the notion of 'almost same time' requires a time-window duration to be specified. In this paper, we use this as a tuning parameter: typically 15 minutes. Further, in the context of 'almost the same resources', the objective is to focus on the bottleneck sections and group the trains for better utilization of primarily these resources.

Dailyzing is the process of grouping trains into one daily path: since the train-time-data consists of time-values across different days (and hence beyond 24 hours), one needs to perform a 'modulo operation' to divide by 24 hours and get the remainder. (Being a large rail-network, some trains' traversal times cross 48 hours too.) The modulo operation maps the set of trains to allotted daily paths. Once the mapping is performed, clustering is used on that set of trains. Clustering is a method of unsupervised grouping of similar objects. The grouping is 'unsupervised' since in our work, the data is not pre-labeled, and there is no defined dependent variable. This type of unsupervised learning is usually done to identify similarity-patterns in the data and to group similar data.

With the objective of efficient clustering of non-daily trains using GQD train data, various clustering techniques have been explored in this paper: see Table 1 for the list of techniques we pursue in this paper. We briefly describe them below.

Table 1: Clustering techniques typically used in such problems

Sr.Num.	**Clustering techniques**
1.	Hierarchical Agglomerative Clustering (HAC) [7]
2.	BIRCH [11]
3.	DBSCAN [4]
4.	Spectral Clustering [8]
5.	K-means [5]

The technique K-means is an elementary and fast clustering method. It requires a pre-defined cluster number (usually called K, and hence the name), which is then used with the distance metric for data clustering. The algorithm initialize by randomly forming groups and their centroids are calculated then distance of each data points from the centroids are calculated and data points with higher distance get reassigned to other groups. The algorithm keeps iterating till groups are formed with minimum group variance. The K-means method sometimes has a difficulty in repeatability and consistency of results since the initial/starting cluster could possibly play substantial role in the end outcome.

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. The main idea of DBSCAN is that clusters are high density regions in data space and can be separated from low point density regions. The method does not require pre-defined cluster number information in the beginning. It has an epsilon hyper-parameter to help identify the neighbor of any data points and a threshold of minimum number of points for cluster to be called dense. The algorithm randomly picks points in the set and if within epsilon radius of that point, there is at-least threshold number of points, then all points are part of the same cluster. The cluster grows by repeating calculation on each neighboring points. It works for any data size and shape. However, it is known to sometimes perform unsatisfactorily with data of heterogeneous density clusters.

BIRCH stands for Balanced Iterative Reducing and Clustering using Hierarchies. This method works well on large data sets by forming a small representative set of a larger one, but it works only for data sets with metric attributes, and no categorical attributes are utilizable by the technique.

Spectral clustering has roots in graph theory and linear algebra. This technique uses eigenvalues and eigenvectors of the Laplacian matrix of the graph for association and clustering. This method is primarily meant for graph data, but it is also known to work satisfactorily on general data.

Hierarchical Agglomerative Clustering (HAC) starts with each data point as a cluster and agglomerate (merge) the data points to reduce the number of clusters. Hierarchical Agglomerative clustering is elaborated in Section 3.

In this paper, we pursue a comparative study of various types of clustering techniques: centroid based clustering K-means, density based clustering in noise: DBSCAN, and connectivity based clustering using hierarchical: HAC in terms of the number of conflict-free cluster formations and total execution time of these clustering methods. For the data that we work with, HAC gives the best result for the application we consider.

2. Problem statement

Though mentioned briefly in the previous section, in this section, we elaborate on the problem dealt within this paper. Usually, in any rail-network, trains have a weekly pattern, and many trains are non-daily. However, even for the non-daily trains, having a daily path for these non-daily trains is common. Having daily paths is essential to have day-independent maintenance blocks. Further, freight paths can be scheduled each day by utilizing the un-utilized time-windows. Of course, over and above what is timetabled for freight trains or scheduled for maintenance activities, unused paths of certain days belonging to non-daily trains (but not running on that week-day) can also be used additionally for freight trains/maintenance.

2.1. Dailyzing Process

For the purpose of this work, we use the phrase: **dailyzing** in the following sense. It is the process of grouping trains into one daily path: this maps the train set to allotted daily routes. Thus if "dialyzing" is insufficiently pursued, then compacting is insufficient, and hence too many paths are scheduled. Due to this, the lower-priority trains would find more difficulty during the timetabling process: especially in the bottleneck sections.

2.2. Dailyzing objective

We focus on the following objective in this paper: to group the non-daily trains efficiently based on their current schedule to have a representative daily train of the group: just this representative would be utilized for future scheduled process and thus leading to the scheduling of the whole group. Further, the benefits/tasks are as follows:

1. Faster timetabling procedure: grouped non-daily trains can be represented by a daily train, scheduling which automatically schedules others in the group (for the common/overlapping sections: the overlapping sections often being the bottleneck sections too).
2. Possibility of new trains: clusters with fewer than seven members can accommodate more non-daily trains. Hence, availability of time-slots with the purpose of introducing new trains.
3. Efficient clustering would help in determining under-utilized and over-utilized resources.
4. Suggestion for better timetabling, i.e. rescheduling some trains could lead to better compaction and efficient resource utilization.

3. Background and related work

Literature is rich with work on the train timetabling problem (TTP). There is work on periodic scheduling and non-periodic scheduling, single one-way line and general rail network TTP, programming and heuristics based TTP, robust and maximum efficiency TTP [1]. In this paper we are not generating the timetable from scratch, rather we are using draft timetable data to improve upon the timetabling procedure and tweaking only some trains to improve efficiency and resource utilization. Recent work using clustering to improve upon railway operations has been by [2]: that work uses K-means clustering to recognise recurrent behavioral delay pattern in railway signal system. Using those clusters this paper identifies where it is taking place and suggests necessary corrective actions. Hartigan et.al [3] uses clustering for high-speed railway trains classification into three category and suggests some operational changes for good operational performance. Wenxian et.al [9] proposed affinity propagation based automatic classification of passenger flow interval for adaptive line planning of high speed train to meet demand. Chang'an et. al [10] have used principle component analysis to reduce attributes of railway station then they used Hierarchical clustering for station classification for train operation planning.

The rest of this section consists of details about the best performing clustering method i.e., Hierarchical Agglomerative clustering (HAC).

Hierarchical Clustering performs grouping the data based on idea that nearby objects are more related than far ones. Its connectivity based clustering and performs grouping with an appropriate measure of 'similarity'. It fits our clustering problem, i.e. clustering using a similarity matrix. Given a set of N items to be clustered and an $N \times N$ distance (or alternatively, a similarity) matrix, the basic process of the hierarchical clustering [7] is as follows:

1. Start by assigning each item to a cluster so that if we have N items, now we have N clusters, each containing just one member. Let the distances (similarities) between the clusters be the same as the distances (similarities) between the items they have.

2. Find the closest (most similar) pair of clusters and merge them into a single cluster so that now we have one cluster less.

3. Linkage: compute distances (similarities) between the new cluster and each of the old clusters.

4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N or stop detecting low cluster cohesion.

Step 3 can be performed in different ways based on the notion of 'linkage': we list three of them: single-linkage, complete-linkage, and average-linkage clustering. In ***single-linkage clustering***, one considers the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. Whether a data-point should belong to one cluster or another gets decided by the highest similarity value from any member of one cluster to any of the members of the other clusters. This technique is also called the connectedness or minimum method.

Alternatively, in the so-called ***complete-linkage clustering*** one considers the distance between one cluster and another cluster to be equal to the greatest distance from any member of one cluster to any member of the other cluster. This linkage is also called the diameter or maximum method. Finally, in the ***average-linkage clustering***, one defines the distance between one cluster and another cluster to be the average of all the pair-wise distances between all members of one cluster and all members of the other cluster. This kind of hierarchical clustering is called Agglomerative, as it merges clusters iteratively. HAC allows complex shape cluster formation.

4. Method employed

In this section, the train-running data we use and the methods applied to it are elaborated. The data used is from an important rail-subnetwork: so-called the Golden Quadrilateral and the Diagonals (GQD) of the Indian Railway network. The data structure, routes' information, and data size routewise is mentioned first, and then various preprocessing steps are explained. After that, the calculation of distance metric is mentioned, followed by various HAC hyperparameters tuning.

4.1. Golden Quadrilateral and Diagonals (GQD) Data

The GQD is an important subnetwork of the Indian Railway network. It is a multi-route railway network. GQD data routes map, sample data, and data size routewise are shown below. The quadrilateral being formed between the four major cities within India is a high-rail-traffic route and, together with the diagonals, form an important high-density network: Figure 2.

A sample of the GQD data taken from a draft timetable, dated around August 2020 is shown in Table 2. The various terms are described in Table 3: all the proposed clustering techniques use the data as per these formats.

Table 4 describes the GQD routewise data size that we use in this paper.

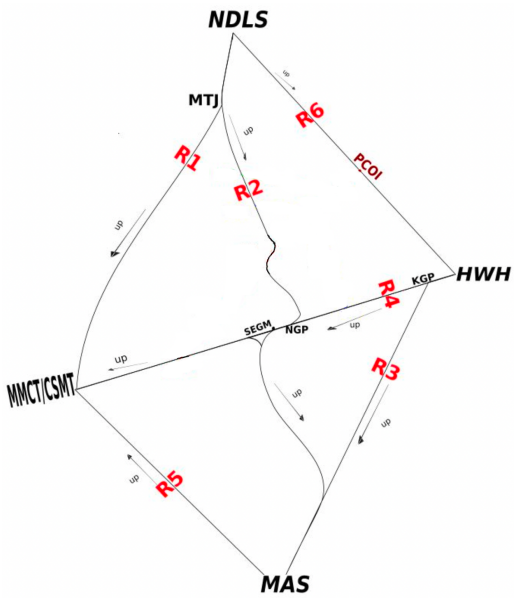

Figure 2: The Golden Quadrilateral and Diagonals (GQD): route map

Table 2: Sample GQD dataset

TRNNUM	WEEKDAYS	STN CODE	ARVL	BLCKSCTN	DAY	Direction
22922	0,1,0,0,0,0,0	CNB	44400	CNB-CPA	1	up
22922	0,1,0,0,0,0,0	MTJ	69600	MTJ-BSA	1	up
22922	0,1,0,0,0,0,0	BTE	76800	BTE-SWAR	1	up
22922	0,1,0,0,0,0,0	SWAR	77520	SWAR-JCU	1	up
22922	0,1,0,0,0,0,0	JCU	77700	JCU-PNGR	1	up

Table 3: GQD dataset terms description

TRNNUM	denotes train number
WEEKDAYS	'1' denotes that the train runs on that day of the week (with the first '1' as Sunday).
STATION	denotes the station through which the train passes.
ARVL	arrival time of the train at the given station.
BLCKSCTN	is the block-section between two stations.
DAY	day of the journey of the train when it commenced from the source.
Direction	is the direction of the train depending on the route's notion of up/down.

Table 4: GQD routewise data size values

Route number	**Total number of trains**	**Number of daily trains**	**Number of non-daily trains**	**Number of block-sections**	**Number of stations**
1	567	312	255	394	198
2	562	319	243	600	301
3	360	210	150	560	281
4	462	268	194	586	294
5	332	220	114	408	205
6	641	401	240	506	254

4.2. Data preprocessing and further definitions

The following preprocessing steps are performed on train data before it is ready for clustering purposes:

1. Route-wise split of all trains: if a train runs on multiple routes during its journey, then the train's journey is split into sub-trains based on routes.
2. Removal of **daily Trains** from data: identifying all daily trains in the GQD data and removing these trains' data, thus leaving only non-daily train data.
3. Single day mapping: train-time data needs to be converted to the same-day, since the time-value of an event often is beyond 24 hours. Since there are 86400 seconds in a day, we perform the 'modulo 86400' operation to the timings of all non-daily trains spread throughout the week and thus the event-times get mapped to a single day event-time. This helps comparing train timings for grouping purposes.
4. Removal of so-called **single touch trains** in a given route, i.e. the trains with only one station in the route: these trains' data are discarded.
5. Detection/removal of so-called 'Geo-loops' in route: when the train comes back to the same station/block-section after a small diversion, we call that part of the route a Geographical-loop or Geo-loop. We remove the Geo-loop part of the journey from our data before processing further.
6. Classifying each block-section into **up/down** direction that is predecided for each route: all block-sections.

After data preprocessing, the distance metric is calculated for clustering in the following section.

4.3. Distance metric: similarity/dissimilarity matrix

The calculation of the distance metric (similarity/dissimilarity matrix) is crucial. A good and meaningful distance metric leads to better clustering. We generate three metric matrices for the purpose of clustering. The first one is a similarity matrix. We call it the Cluster Similarity Matrix (CSM). This matrix holds information about the similarity of trains based on their running, first on a particular block-section at a particular time, and then eventually the matrix holds information of similarity between every pair of trains accumulated over all common block-section between those trains. The second matrix is Normalized Cluster Similarity Matrix (NCSM). This matrix is just the normalized version of the CSM. The third and last one is Normalized Cluster Dissimilarity Matrix (NCDM) or distance matrix. Further details of them are below.

Definition 4.1. *The similarity index (or Similarity score) is a distance metric defined on trains based on their closeness in the distance vs time chart. Two trains are said to be of high similarity index if they arrive at the same block-section at almost the same time. Refer to Equation* (1) *and Figure 3.*

In order to be grouped together, trains getting higher similarity index amongst each other are required to be 'complementing' in days of a week, i.e. they must run on different days of a week on the same block-sections at the approximately same time (refer to Definition 1.3).

1. **Cluster Similarity Matrix (CSM):** It is an $N \times N$ matrix where N is the number of non-daily trains operating on that route. The rows and columns of this matrix are indexed by trains. The similarity matrix is generated by matching each pair of trains at a given block-section to get a similarity index using equation (1). We get the i-j^{th} entry of the similarity matrix by accumulating such similarity across block-sections and normalizing it across the number of block-sections. Trains running at the same block-section at the same exact time would get a similarity index value of 1. If one of them arrives beyond a chosen threshold duration from the previous train, the time window (taken as 1500 seconds), then they get a similarity value of 0.

$$\text{SimilarityScore} = \begin{cases} \cos\left(\dfrac{\pi(\mathrm{T}_i - \mathrm{T}_j)}{2 \times \text{Time\_Window}}\right), & \text{if Time\_Window} > |\mathrm{T}_i - \mathrm{T}_j|, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathrm{T}_i, \mathrm{T}_j$ are the times of arrival for the Train i and Train j respectively, at a given block-section, and Time_Window is the allowed time within which two trains are considered to be similar: this is taken as 1500 seconds. Next, the above

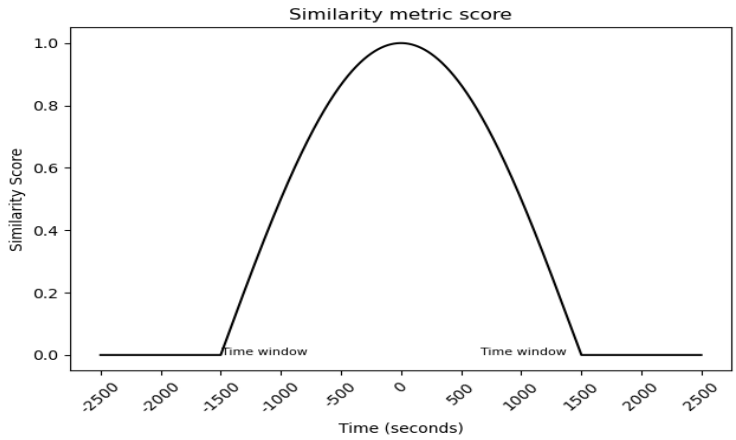

Figure 3: Similarity index calculation.

score is aggregated across all block sections (and, as explained in the next section, normalized by dividing the matrix by the number of block sections used aggregation.) Further, for the computation of cosine of an angle, the argument of the cosine function above is viewed to be specified in radians. The CSM is obtained by accumulating the similarity matrices over all block-sections. CSM is a symmetric matrix.

2. **Normalized Cluster Similarity Matrix (NCSM):** This is a normalized CSM. The matrix is normalized across rows and columns to get diagonal values equal to 1. The rows and columns of this matrix are indexed by trains. All values are between 0 & 1 and higher values means more similarity. Of course, its diagonal elements are all 1's and the matrix is symmetric.
   For example, in the matrix shown below value 0.1 in the 1^{st} row & 3^{rd} column means relatively low similarity between Train 1 and Train 3: $\begin{bmatrix} 1 & 0.5 & 0.1 \\ 0.5 & 1 & 0 \\ 0.1 & 0 & 1 \end{bmatrix}$
3. **Normalized Cluster Dissimilarity Matrix (NCDM):** The dissimilar matrix is a measure of 'distance' that we want to minimize when choosing a cluster. NCDM is obtained by subtracting the NCSM from the all-ones-matrix A, i.e. A is square matrix of appropriate size with each entry as '1'. The rows and columns of this matrix are indexed by trains. All values are between 0 & 1 and higher values means more dissimilarity. Its diagonal elements are all 0's. It is, again, a symmetric matrix.
   For example in matrix shown below value 0.9 at 1^{st} row & 3^{rd} column means higher dissimilarity between Train 1 and Train 3: $\begin{bmatrix} 0 & 0.5 & 0.9 \\ 0.5 & 0 & 1 \\ 0.9 & 1 & 0 \end{bmatrix}$

Once a normalized dissimilar cluster similarity matrix is generated, Hierarchical Agglomerative Clustering (HAC) is used to form clusters of similar trains.

4.4. Choice of optimal number for number of clusters

There are different approaches to choosing an optimal number of clusters or when to stop combining smaller nearby clusters into one. The diameter of the cluster or radius of the cluster is used as the threshold to detect low cluster cohesion and deter cluster combining. Silhouette score is also used for the same. We use the following approach.

We use the notion of 'singular' cluster for one that has just one or two members. The **best fit** is found by **maximizing** the number of non-singular clusters for a total number of clusters formed (see Figure 4).

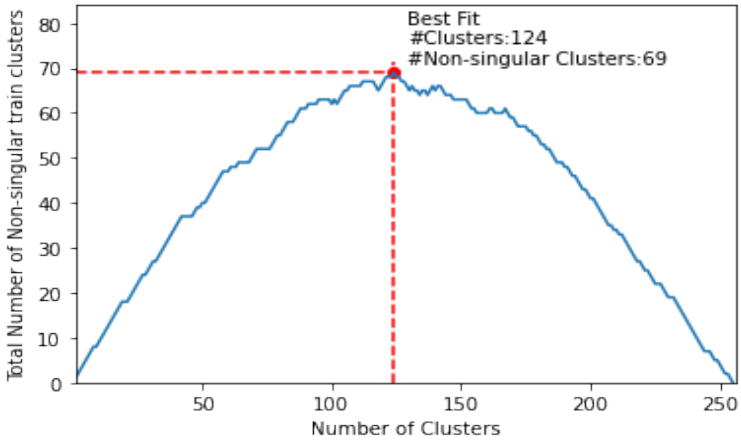

Figure 4: Choosing an optimal number of clusters.

4.5. Hyper-parameter tuning of HAC

Linkage decides ways on distance metrics calculation to assign a member to cluster; refer to 3. Affinity is a distance metric. Just like linkage, there are various distance metrics such as Euclidean (distance between the points), Manhattan (sum of absolute values of difference of points), cosine (angle between the points), and precomputed (user-defined distance matrix is used). Choosing the best linkage and affinity for Agglomerative clustering is done based on the number of clusters formed and the size of the clusters. Tables 5 and 6 show the data suggesting average linkage and precomputed affinity for HAC.

Table 5: Hierarchical Clustering (Ward linkage and variable time window)

Route number	Total unique trains	Total clustered trains	Number of clusters	Maximum number of trains in cluster	Total number of conflicting cluster
1	257	172	66	6	3
2	256	151	57	6	4
3	166	105	32	7	2
4	194	132	47	7	1
5	123	65	25	5	3
6	251	158	56	7	3

Table 6: Hierarchical Clustering (average linkage and variable time window based on station distance)

Route number	Total unique trains	Total clustered trains	Number of clusters	Maximum number of trains in cluster	Total number of conflicting clusters
1	257	187	63	6	5
2	256	151	55	6	4
3	166	98	33	6	3
4	194	129	41	7	0
5	123	79	27	7	3
6	251	155	57	6	1

Similarly, another hyper-parameter “time window”, is chosen. Time window could be an optimal “fixed” time window or “variable” time window based on block-sections size. Tables 6 and 7 provide an answer about how to choose one based on the number of clusters formed, the size of individual clusters, and the number of conflicting clusters it produced.

Table 7: Hierarchical Clustering (average linkage and optimized fixed time window)

Route number	Total unique trains	Total clustered trains	Number of clusters	Maximum number of trains in cluster	Total number of conflicting clusters
1	257	208	67	7	7
2	256	184	60	7	10
3	166	103	34	6	2
4	194	138	48	7	2
5	123	71	26	5	3
6	251	162	58	7	2

5. Results

During every hyper-parameter tuning, such as time window (fixed or variable time window based on block-section sizes), the cluster linkage method and affinity, conflicting clusters are formed, i.e. clusters not satisfying complementing condition (see Tables 5, 6 and 7). Hence, the next objective is to maximize the number of conflict-free train clusters.

Table 8: Conflict-free clusters of non-daily trains across routes formed using HAC

Route number	Total unique trains	Total conflict free clustered trains	Total number of clusters
1	257	208	77
2	256	178	64
3	166	123	43
4	194	146	54
5	123	73	30
6	251	198	57

Table 8 shows the number of trains clustered into conflict-free clusters and the number of clusters formed using HAC for GQD data: routewise.

Table 9: Comparative study of different clustering techniques and their execution time

		HAC		DBSCAN		K-means	
Route number	Total unique trains	Number of conflict free clustered trains	Time taken (sec)	Number of conflict free clustered trains	Time taken (sec)	Number of conflict free clustered trains	Time taken (sec)
1	257	208	2.15	173	13.70	198	312.70
2	256	178	3.19	134	19.03	179	458.18
3	166	123	1.37	85	5.79	115	142.20
4	194	146	2.52	132	5.38	150	224.95
5	123	73	0.85	57	1.66	78	82.15
6	251	198	3.74	167	16.12	190	313.70

Table 9 shows a comparative study of three clustering techniques: HAC, DBSCAN & K-means by comparing their execution times and the number of trains gets clustered. The number of clustered trains shown here are all conflict-free, i.e. they satisfy complementing condition: see Definition 1.3.

6. Conclusion and further directions

We have used unsupervised methods to cluster ‘similar’ trains based on their similarity index (see Definition 4.1). The dissimilarity index calculated from the similarity index gives us a sense of distance, which is then used to cluster similar trains together by minimizing the distance matrix. GQD data, after preprocessing and using appropriate similarity/dissimilarity metrics, is clustered nicely. The generated clusters match actual running train data. The clusters developed also complement each other on different days of the week very well, with very few failing to complement and form conflicting groups.

The time required from the pre-processing of the train data till the time we generate clusters using HAC is within 3-4 seconds for all routes data in GQD. This time is very less compared to K-means and DBSCAN clustering techniques' execution time; see Table 9. Hence, HAC is faster and more efficient time wise and thus HAC suits the application. The train clusters obtained will be tested in mixed-rail traffic simulator for performance and validation. Also the suggestion will tested in simulator for actual performance enhancement validation.

A possible future direction is to include the complementing condition into the clustering process, rather than ruling out already-formed clusters that have a conflict. Another useful task to be pursued is to list out those non-daily trains that do not fit well into any cluster and thus need a tweak in their timings and thus help free-up a daily-path (usable for faster freight trains or a new-introduction.)

References

[1] Cacchiani, V., Galli, L. and Toth, P., A tutorial on non-periodic train timetabling and platforming problems, *EURO Journal on Transportation and Logistics*, 2192-4376, 2015.

[2] D'Ariano, A., Cerreto, F., Nielsen, B. F., Nielsen, O. A., Harrod, S. S., Application of Data Clustering to Railway Delay Pattern Recognition, *Journal of Advanced Transportation*, Hindawi, 0197-6729, 2018.

[3] Deng, L., Chen, Y., Wu, R., Wang, Q. and Xu, Y., A two-dimensional clustering method for high-speed railway trains in China based on train characteristics and operational performance, *IEEE Access*, vol. 8, pages 81918-81931, 2020.

[4] Ester, M., Kriegel, H.P., Sander, J. and Xu, X., A density-based algorithm for discovering clusters in large spatial databases with noise, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press*, pages 226–231, 1996.

[5] Hartigan, J. A., and Wong, M. A., Algorithm AS 136: A K-means clustering algorithm, *Journal of the Royal Statistical Society*, Series C. 28 (1): 100–108, 1979.

[6] *Indian Railways Year Book 2020-2021*, Ministry of Railways, Government of India, New Delhi, 2022.

[7] Johnson, S. C., Hierarchical clustering schemes, *Psychometrika*, vol. 32, pages 241-254, 1967.

[8] Ng, A., Jordan, M. and Weiss, Y., On spectral clustering: analysis and an algorithm, *Advances in Neural Information Processing Systems 14*, pages 849–856, MIT Press, 2002.

[9] Wang, W., Shi, T., Zhang, Y., and Zhu, Q., An affinity propagation based clustering method for the temporal dynamics management of high speed railway passenger demand, *Journal of Advanced Transportation*, 2021.

[10] Xu, C., Li, J., Zou, C., and Ni, S., Application of Hierarchical Clustering Based on Principal Component Analysis to Railway Station Classification, *Sixth International Conference on Transportation Engineering*, 2020.

[11] Zhang, T., Ramakrishnan R., and Livny, M., BIRCH: an efficient data clustering method for very large databases, *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 103-114, 1996.