Application of Graph Theory in Air Traffic

A Dissertation for

MAT-651 Discipline Specific Dissertation

Credits: 16

Submitted in partial fulfilment of Masters Degree

M.Sc. in Mathematics

by

Mr. ADAN JOAQUIM FERNANDES

22P0410001

ABC ID : 967-310-592-918

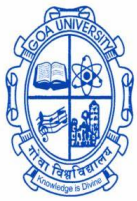
201900252

Under the Supervisor of

Dr. JESSICA FERNANDES E PEREIRA

School of Physical & Applied Sciences

Mathematics Discipline



GOA UNIVERSITY

APRIL 2024

Examined by: Seal of the School

DECLARATION BY STUDENT

I hereby declare that the data presented in this Dissertation report entitled, "Application of Graph Theory in Air Traffic" is based on the results of investigations carried out by me in the Mathematics Discipline at the School of Physical & Applied Sciences, Goa University under the Supervision of Dr. Jessica Fernandes e Pereira and the same has not been submitted elsewhere for the award of a degree or diploma by me. Further, I understand that Goa University will not be responsible for the correctness of observations / experimental or other findings given the dissertation.

I hereby authorize the University authorities to upload this dissertation on the dissertation repository or anywhere else as the UGC regulations demand and make it available to any one as needed.

Signature: 

Student Name: ADAN JOAQUIM FERNANDES

Seat no: 22P0410001

Date: MAY 08, 2024

Place: GOA UNIVERSITY

COMPLETION CERTIFICATE

This is to certify that the dissertation report "Application of Graph Theory in Air Traffic" is a bonafide work carried out by Mr. ADAN JOAQUIM FERNANDES under my supervision in partial fulfilment of the requirements for the award of the degree of Master of Science in Mathematics in the Discipline Mathematics at the School of Physical & Applied Sciences , Goa University.

Signarure : _____

Supervisor : Dr. Jessica Fernandes e Pereira

Date: MAY 08, 2024

School of Physical and Applied Sciences Goa University

School Stamp

Signature of Programme Director of Mathematics

Date: 10/5/2024

Place: Goa University

PREFACE

This Project Report has been prepared in partial fulfilment of the requirement for the Subject: MAT - 651 Discipline Specific Dissertation of the program M.Sc. in Mathematics in the academic year 2023-2024.

The topic assigned for the research report is: "Application of Graph Theory in Air Traffic ". In this survey, each section of the chapter uses a graph theory based algorithm to solve the problem of graph partitioning, and find partitions of the graph with respect to the given objective.

FIRST CHAPTER :

The Introductory stage of this Project report is based on overview of the DAC (Dynamic Air Configuration) problem, which deal with the formation of new sectors of a given air network, with the presence of different constraints which effect the formation of the flight paths, sectors and network.

SECOND CHAPTER:

This chapter deals with DAC method using which the data of air network and sectors to form a weighted graph, then the graph partitioning of the weighted graph is then done with the help of Voronoi diagrams to form a new weighted graph according to required objectives. The k-means algorithm is then used to partition the graph into subgraphs and finally the weights are balanced to reduce the workload of each sector.

THIRD CHAPTER:

The paper presents a new method for DAC which is based on a weighted graph model

by applying GWGC algorithm and ODLB algorithm in combination with a Heuristic algorithm to partition a given airspace into sectors while also taking in accounts different parameters and objectives.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my Mentor, Dr. Jessica Fernandes e Pereira, who was a continual source of inspiration. She pushed me to think imaginatively and urged me to do this homework without hesitation. Her vast knowledge, extensive experience, and professional competence in Graph Theory enabled me to successfully accomplish this project. This endeavour would not have been possible without her help and supervision.

ABSTRACT

Air travel being a prominent mode of travelling we have an increased number of flights travelling each day, with the increase in flights there is also an increase in casualties, time delay and other issues that have made air traffic control and management a very crucial part of air traffic. There is a need of managing the air traffic in real time by changing routes, while maintaining the air traffic so as to minimize the casualties and other problem faced during the flight while also using the routes optimally with a safe network and inturn improving the service provided.

The main aim of this article is to study the application of graph theory in DAC (Dynamic Air Configuration), study the methods to create a new flight network within a given time frame and with the data provided to us, methods to find optimal edge cut to improve the workload balance between airports is also studied, with the use of graph partitioning on weighted graphs, using the likes of voronoi diagrams to form different sectors these sectors are used to form new subgraphs using the K-means algorithm.

Keywords: Air traffic, DAC, Weighted graphs, Edge cut, Voronoi diagram, graph partitioning, sectors, K-means algorithm.

Contents

List of Figures

Notations and Abbreviations

$V(G)$	vertex set of G
$E(G)$	edge set of G
G_w	weighted graph G
G_w^i	partitions of G
w_i	P_3 vertex weight of G_w
w_{ij}	edge weight G_w
$w(G_w^i)$	sum the vertex weights in G_w^i
w	vertex weight vector
W	edge weight matrix
C_i	cells of G

Chapter 1

INTRODUCTION

In air traffic management since there are only several controllers it becomes impossible to put all air crafts under surveillance, flying in the whole air space of a nation, the airspace is usually divided into smaller regions known as sectors, and each sector is under the observation of one or more controllers. In this manner the air craft count is not supposed to exceed the controllers ability to monitor.

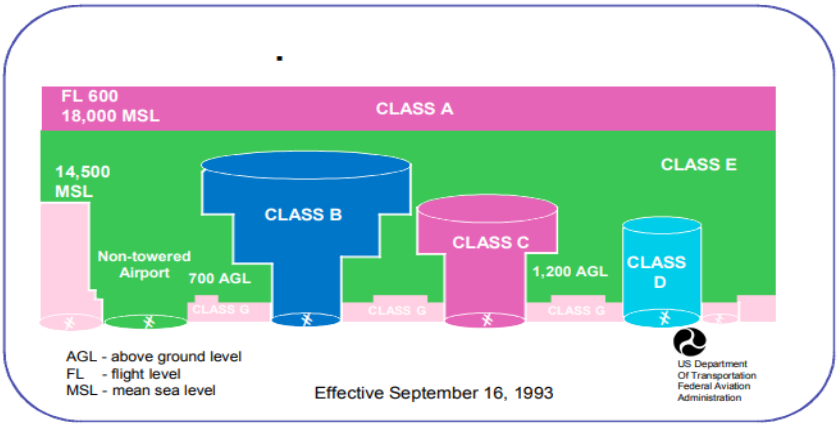


Figure 1.1: Sectors in Air Space

The configuration of the fixed sectors correspond to the way that relatively few aircraft fly along the fixed air routes. The airspace characterized by fixed air routes and fixed sectors is referred to as a structured and static one. With the improvement and innovation in air traffic, routes structure and demands have changed over the years, while a big number of flights traverse through the same routes simultaneously at different flight levels, there is an increase in the number of flight delay and they arise due to various reasons such as bad weather and traffic congestion. The solution to this problem can be attained partly by improving the method in which the flight changes its air routes, instead of following fixed routes during the flight at all times. Consequently the sector counts and boundary vary with traffic change. This is a dynamic airspace configuration problem (DAC).

Dynamic airspace configuration (DAC), is an encouraging concept proposed to convert airspace sectorization from the structured and static airspace to a dynamic one capable of accommodating dynamically changing traffic demand. A lot of research on DAC has been carried out, and most scholars completed DAC by describing the airspace as a model and then adopting a proper algorithm to partition the airspace into sectors.

Due to the graph model being embedded with information on underlying topological structure of the airspace, it usually helps to consider the factors such as air routes and key points, i.e., airports, crossing points as well as waypoints for DAC. Therefore, the graph model is preferred in this study [1]. Furthermore, we also consider traffic flows along air routes which are used to compute the workloads. The workloads can be assigned as the edge weights and the vertex weights. Such topological structure with traffic flows can be described as a weighted graph mathematically. Thus, the weighted graph model is adopted for DAC here. And it is different from the traditional weighted graph that only edges are assigned with weights, here we obtain an undirected graph with the weights on both vertices and edges, where traffic information incorporated. This is the key feature of our graph model. The airspace has several constraint that are taken into account. These

constraints must be used while forming the sectors.

The first is workload constraint. The constraint points out that the workload of each sector should be below a threshold and the workloads of those sectors are balanced, it also ensures that workload of each sector does not exceed the controller's capacity to control the aircrafts while the workloads are evenly distributed among designed sectors. The second is geometric constraints consisting of the route based convexity constraint, connectivity constraint and minimum distance constraint. The convexity indicates that an aircraft should not enter the same sector twice.

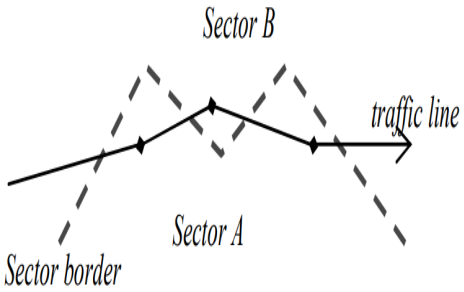


Figure 1.2: Convexity Constraint

The connectivity constraint is that a sector is not fragmented.

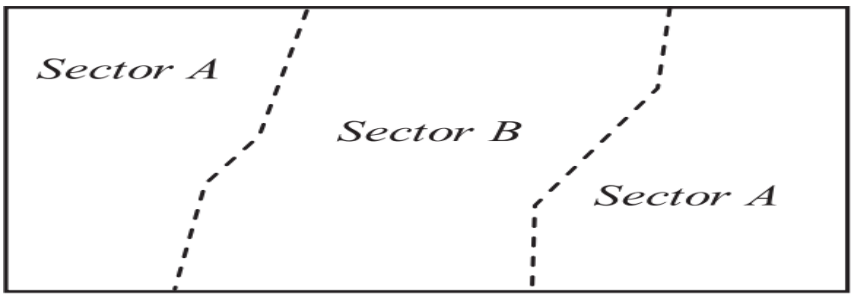


Figure 1.3: Connectivity Constraint

The minimum distance constraint means that the distance between the sector boundaries and the key points as well as the distance between the boundaries and the air routes is not less than a given minimum value.

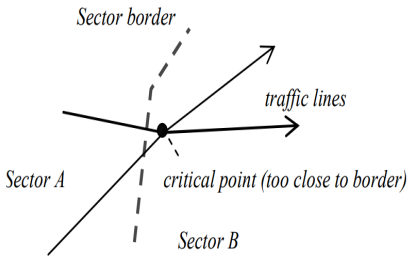


Figure 1.4: Distance Constraint

The geometric constraints ensure that the controller have adequate time to control the aircraft and to solve conflicts which may happen. These constraints are critical to ensure the safety of aircrafts. Hence, the above constraints are considered thoroughly in the study [3].

There are a several metrics for workload such as traffic mass, aircraft count, dynamic density, and so on [1]. Computing workload metrics other than aircraft count might have taken more factors into accounts. However, there is no evidence that Traffic Mass and dynamic density are more effective than aircraft count for DAC. Workload metric other than aircraft count might be prohibitive in practical application. Thus, aircraft count is adopted as workload metric in this study.

DAC method based on a weighted graph model is developed in the study. We set up a weighted graph model for a given airspace which accurately describes the airspace structure information and traffic data. The procedure begins with constructing an undirected graph model for the given airspace, of which the vertices represent the key points such as airports, waypoints, and the edges represent the air routes. [2] The, vertices are then used as the sites of Voronoi diagram which divides the airspace into units called cells, and aircraft counts of both cell and air route are computed. By assigning both the vertices and the edges with those aircraft counts, a graph model is built up. Furthermore, in order to facilitate the discussion, the graph model is simplified into a weighted graph model whose vertices have a one-to-one relationship with Voronoi cells. Accordingly the

airspace configuration problem is described as a weighted graph partitioning problem. Furthermore the paper develops a graph partitioning algorithm that divides the weighted graph model into sub-graphs. The algorithm mixes general weighted graph cuts (GWGC) algorithm, with optimal dynamic load balancing (ODLB) algorithm in the following manner. After the Cuts Algorithm partitions graph model into sub-graphs, the load balancing algorithm together with the Heuristic algorithm transfers aircraft count to achieve workload balancing among the sub-graphs. Finally, the cells corresponding to each sub-graph are combined together into a sector. Besides, the method attempts to design the sectors with the objective of balancing workload, minimizing coordination workload as well as satisfying geometric constraints. [3, 7, 6, 4],

1.1 Voronoi Diagram

Voronoi diagrams in general deal with the following problem that for a given set S of n points(sites) in a plane, we try to associate with each point $s \epsilon S$ a region consisting of all the points in the plane that are closer to s than any other point $\tilde{s} \epsilon S$, which in mathematical terms is given by,
Vor(s) = { p: distance(s,p) $\leq$ distance ($\tilde{s}$, p),$\forall \tilde{s} \epsilon S$ } where **Vor**(s) is the voronoi region of a point s.

1.1.1 Voronoi Diagram for Various Cases

When S consists of a single point, the region for the point in the whole plane.
When S consists of two points the Voronoi diagram is given by.

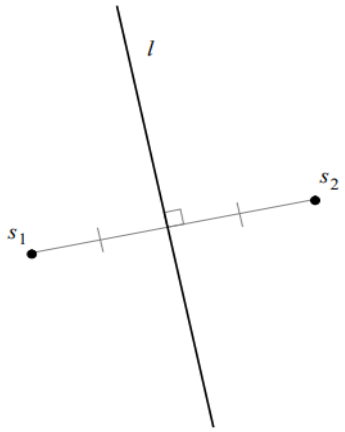


Figure 1.5: Set of Two Points

The Voronoi diagram for the two point set in figure1.5 $S = \{ s_1, s_2 \}$ consists of two half planes divided by the ray, which is the perpendicular bisector of the line segment $\overline{s_1, s_2}$. Note that the two regions are not disjoint, but overlap at the set of points equidistant from both points of the ray. We define the bisection of $\overline{s_1, s_2}$, as the set of all the points $b(s_1, s_2) = \{x \epsilon \mathbb{R} : \|xs_1\| = \|xs_2\|\}$.

Theorem 1.1 [5] All points on the half plane containing s_1 and delimited (having fixed boundaries or limits) by the perpendicular bisector l of $\overline{s_1 s_2}$ are closer to s_1 than s_2.
Proof: Consider a point p in the half-plane containing s_1. We can construct two right triangles $\triangle s_1 pb$ and $\triangle s_2 pb$. They both share a side$\overline{pb}$ and $\overline{s_1 b}$ is shorter than $\overline{bs_2}$ since $\|s_1 m\| = \|s_1 b\| + \|bm\|$ and $\|s_1 m\| = \|s_2 m\|$. The hypotenuse of $\triangle s_1 pb$, $\overline{s_1 p}$ is shorter than $\overline{s_2 p}$ by the Pythagorean theorem. Therefore p is closer to s_1 than s_2.

□

from theorem 1.1 we can conclude that the set of all the points in the half plane of s_1 can be given by$h(s_1, s_2) = \{ x: \|xs_1\| < \|xs_2\| \}$. As shown in figure1.6

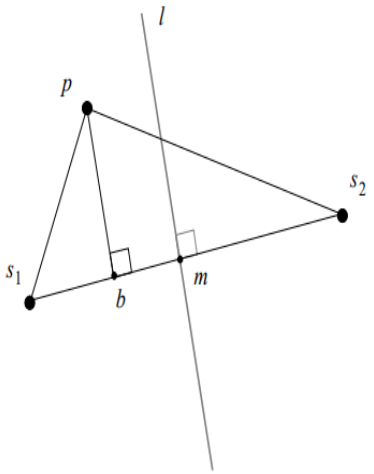


Figure 1.6: Points on a Half Plane

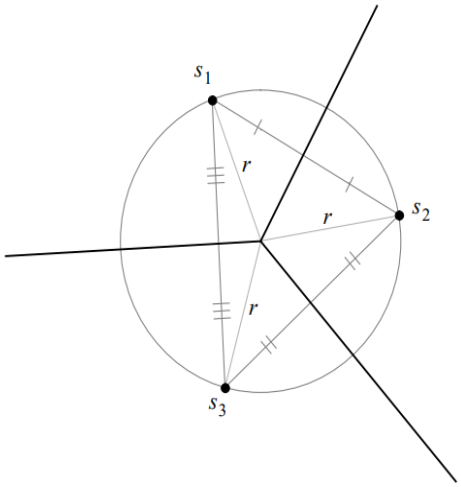


Figure 1.7: Voronoi Diagram for Three Points

The figure1.7 shows a Voronoi diagram for three points, and the geometry used in it's construction. We start by joining each pair of vertices by a line. We then draw the perpendicular bisectors to each of these lines. These three bisectors must intersect, since any three points in the plane define a circle. We then remove the portions of each line beyond the intersection and the diagram is complete. The point where the three rays intersect belongs to the Voronoi regions for all three points. This point is also called the

center of the circle.

Theorem 1.2 [5] The intersection of three perpendicular bisectors of s_1, s_2 and s_3 is the center of the circle containing s_1, s_2 and s_3.

Proof: The point from the perpendicular is equidistant from the two points it bisects. Therefore the segments $\overline{s_1 b}$ and $\overline{s_2 b}$ are equal. Angles $\angle s_1 bc$ and $\angle s_2 bc$ are right angles and both triangles share the side $\overline{bc}$. Therefore the triangles must be congruent and their hypotenuses r_1 and r_2 are equal. A similar argument can be made between s_3 and either of s_1 or s_2, therefore point c is equidistant from s_1, s_2 and s_3 and is thus the center of the circle containing s_1, s_2 and s_3. □

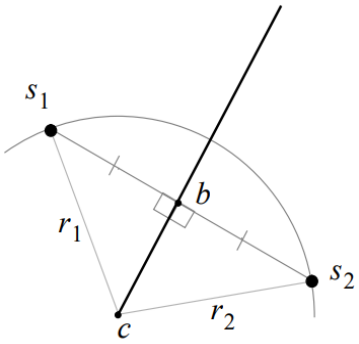


Figure 1.8: Center of The Circle

1.1.2 Voronoi Regions/Cells

There are many methods available to construct a voronoi region for a given point s of the set S. So we begin by taking all of the perpendicular bisectors of the segments connecting s to the remaining members of S. we can use these planes to delimit (set, mark, or draw the boundaries of something) half planes. The intersection of all half planes containing s is the Voronoi region for s, or we can start with the segments connecting s to all remaining

members of S. We then gradually extend lines outward along the perpendicular bisector of these segments until they intersect.

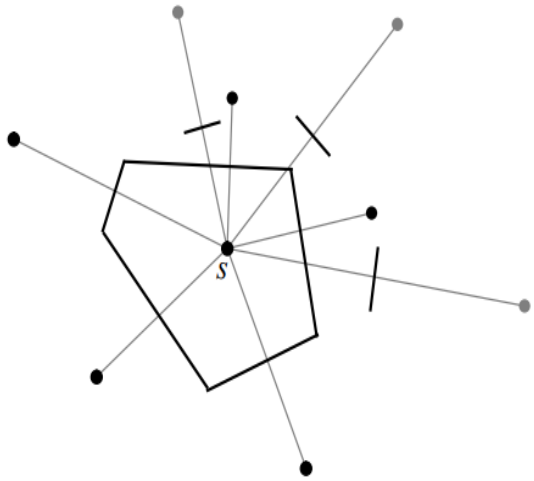


Figure 1.9: Voronoi Region

Note that the points which do not contribute to the region are not necessarily the furthest away, as in figure1.10

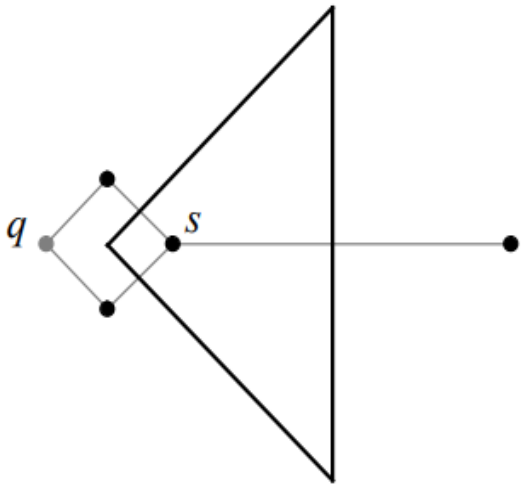


Figure 1.10: Contribution of Sites

1.1.3 A Complete Voronoi Diagram

A Voronoi diagram is the union of all the Voronoi regions in the set: $\mathbf{Vor}(S) = \cup_{s \epsilon S} \mathbf{Vor}(s)$

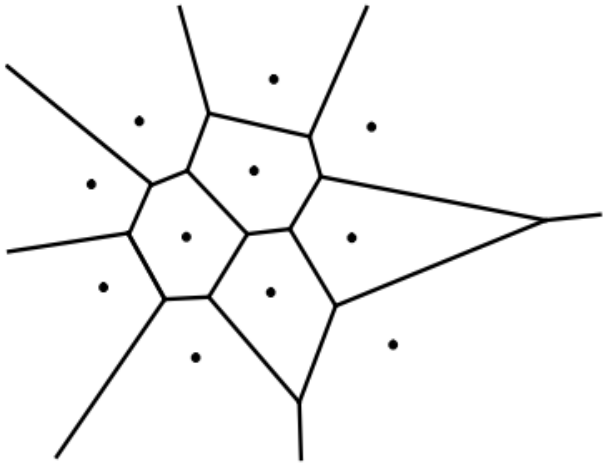


Figure 1.11

The figure1.11 is a Voronoi diagram for a set of 11 points.

1.1.4 Properties of Voronoi Diagrams

Convex hull: A set Y is said to be convex if for any $a, b \epsilon Y$, every point on the straight line segment joining them is also in Y. The convex hull of a set of points X in Euclidean space is the smallest convex set containing X.

Theorem 1.3. [5] For the $Vor(P)$ of a set of points P the following hold.

- **Voronoi vertices**. A point q is a vertex of Vor(P) iff its largest empty circle, called $C_p(q)$ contains three or more sites on its boundary.

- **Voronoi edges**. The bisector between sites p_1 and p_2 defines an edge of $Vor(P)$ iff there is a point q on the bisector such that $C_p(q)$ contains both p_1 and p_2 on its boundary but no other sites.(Then all such points q are on the Voronoi edge).

Proof: For the first property, suppose there is a point q such that $C_p(q)$ contains three or more sites on its boundary. Let $p_1, p_2, .., p_t$ be these sites. Since the interior of $C_p(q)$ is empty point q must be on the boundary of each of $Vor(p_1), Vor(p_2), .., Vor(p_t)$. Hence point q is a vertex of $Vor(P)$.
On the other hand, assume point$q$ is a vertex of$Vor(P)$. Then q is incident to at least three edges, and hence incident to at least thee Voronoi cells $Vor(p_1), Vor(p_2), Vor(p_3)$, $p_1, p_2, p_3 \epsilon P$. Voronoi vertex q is equidistant to p_1, p_2, p_3 and there cannot be another site closer to q. Hence, $C_p(q)$ is an empty circle containing three or more sites on its boundary.

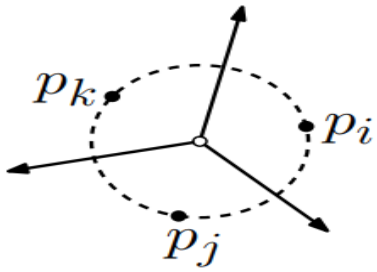


Figure 1.12

For the second property, suppose there is a point q on the bisector between sites p_1 and p_2 such $C_p(q)$ contains p_1 and p_2 on its boundary but no other sites. The dist(q,p_1) = dist(q,p_2) < dist(q,p_x) for any other sit $p_x \epsilon$ P-$\{p_1, p_2\}$. Hence$q$ lies on an edge of$Vor(P)$ that is defined by the bisector of p_1 and p_2. On the other hand let the bisector of p_1 and p_2 define a voronoi edge. Then the largest circle of any point q on this edge must contain p_1 and p_2 on its boundary and no other sites.

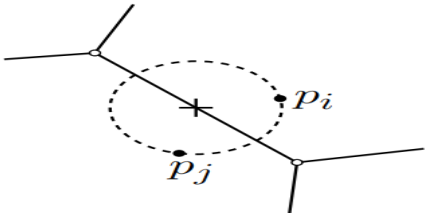


Figure 1.13

Theorem 1.4.[5] The circle containing Voronoi vertex v and passing through the three points s_1, s_2 and s_3 is empty.

Proof: Let s_1, s_2 and s_3 be the three points of S corresponding to the voronoi vertex v. If $C(v)$ contains another point s_4 then s_4 must be nearer to v than any of s_1, s_2 or s_3.In this case v must be contained by the voronoi region for s_4 and not contained in any other region for s_1, s_2 or s_3, by definition of a Voronoi region. However this is a contradiction since v is infact common to the Voronoi regions for s_1, s_2 and s_3.

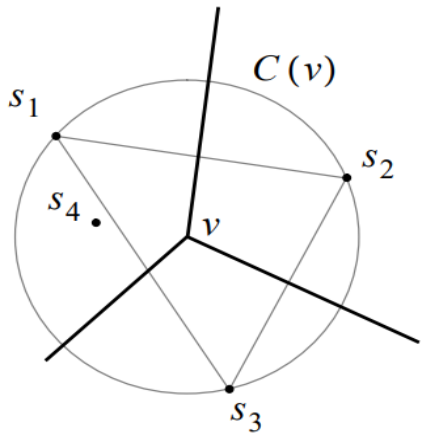


Figure 1.14: Circle passing Through Three Points

1.2 Graph Partitioning/Clustering

Given a weighted graph $G = (V, E, W)$ where V, E are the vertex and the edge sets respectively and W is a weighted adjacency matrix of the graph G.

The graph partitioning problem asks for the subsets of nodes $V_1, ..., V_k$ that partition the node set V, i.e $V_1 \cup ... \cup V_k$ = V, $V_i \cap V_j = \emptyset \; \forall i \neq j$. .
It also asks for partitioning (cut) of the graph into two or more partitions(clusters). Where the size of the cut is the number of edges being cut.
Here an edge that runs between blocks is also referred to as the cut edge.

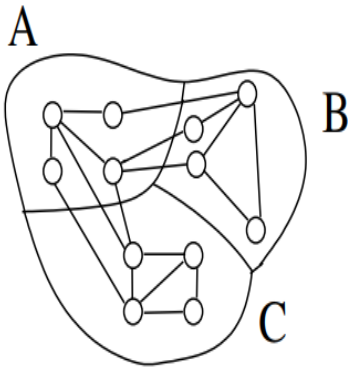


Figure 1.15

here in figure1.15 we have a graph divided into three partitions and each partition has 3 cut edges between (A, B) and (A, C).

Let us denote the $cut(A, B)$ to be the sum of the edge weights between A and B in other words. $cut(A, B) = \sum_{i \epsilon A, j \epsilon B} w_{ij}$.

1.2.1 Weighted Ratio Cut

Minimum Cut Problem: To find a graph partition such that the number of edges between two sets is minimized. We accomplish this by using the following objective. The red cut has 1 cut edge and blue cut has 2 cut edges, which provides us with an unsatisfactory partitioning. To avoid this we use the objective called ratio cut.

Ratio cut: The ratio is a objective which minimizes the edge cut between two graphs. which is given by $RCut(G) = min_{V_1,V_2,....,V_k} \sum_{c=1}^{k} \text{cut}\frac{(V_c,V/V_c)}{|V_c|}$.
for example we have

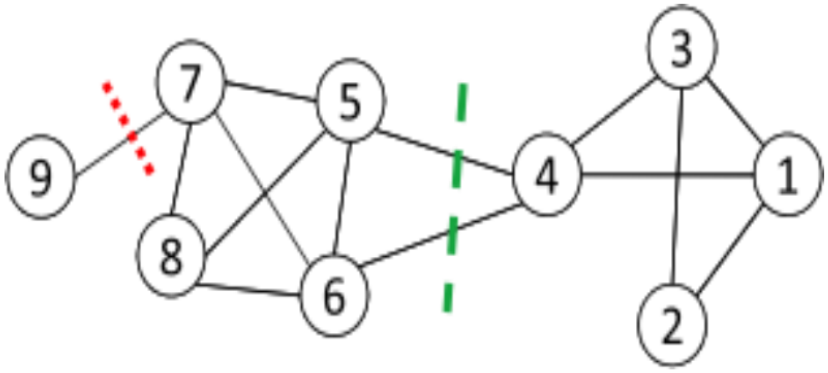


Figure 1.16: Graph With Multiple Cuts

Cut(Red) = 1 edge and the green cut has Cut(Green) = 2 .
Their ratio cuts are given by :
Ratio-Cut(Red) = $\frac{1}{1} + \frac{1}{8} = \frac{9}{8} = 1.125$

Ratio-Cut(Green) = $\frac{2}{5} + \frac{2}{4} = \frac{18}{20} = 0.9$

this gives us the better partition the green cut being the better one in this case, as it has lower ratio cut value.

1.2.2 Matrix Format

We reformulate the objective into a matrix format as follows. Let x_c be an indicator vector for a partition c, i.e x_c= 1 if the partition c contains vertex i and 0 otherwise.
let D = diag$(d_1, d_2, .., d_r)$ be the diagonal degree matrix. obtained by summing the

columns in an adjacency matrix.

The product $x_c^T x_c = |c|$, and $x_c^T (D-A) x_c$ gives us the number of cut edges between c and its complement, where A is the adjacency matrix of the graph G, the matrix $D-A$ is also the Laplacian matrix of the graph and hence $D-A=L$.

we illustrate this using an example below.

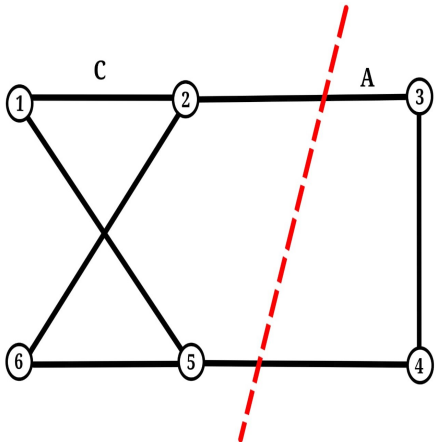


Figure 1.17

We have partitioned the graph in figure1.17 into two clusters, cluster C and A respectively. $x_C^T = [1,1,0,0,1,1]$, $|C| = 4$ i.e the order of c is 4, the number of cut edges in this particular example is 2. we will verify this using matrices. The adjacency matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

, the laplacian matrix $L = D - A = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & 0 & 1 & 3 & -1 \\ 0 & -1 & 0 & 0 & -1 & 2 \end{bmatrix}$

,

the product $x_c^T x_c = 1+1+1+1 = 4 = |C| = 4$ which is the order of the partition C.

$x_c^T L x_c = 2$, which is the no.of cut edges of the partition.

1.2.3 Matrix Form of The Ratio Cut

The ratio cut objective is given by $\min_{V_1,V_2,....,V_k} \sum_{c=1}^{k} \frac{cut(V_c,V/V_c)}{|V_c|}$, from the above definitions we get $x_c^T x_c = |V_c|$, $cut(V_c, V/V_c) = x_c^T L x_c$.

The new ratio cut is given by :$min \sum_{c=1}^{k} \frac{x_c^T L x_c}{x_c^T x_c} = \sum_{c=1}^{k} \overline{x}_c^T L \overline{x}_c$, where $\overline{x} = \frac{x_c}{(x_c^T x_c)^{\frac{1}{2}}}$, Now the above can be written as the minimization of trace$(\overline{X}^T A \overline{X})$, where the c-th column of $\overline{X}$ is x_c.

1.2.4 General Weighted Graph Cuts

The ratio cuts was an objective defined for graphs with edge weights only, now we define the objective for weighted graphs with, vertex weights as well.

$$WCut(G) = \min_{V_1,V_2,....,V_k} \sum_{c=1}^{k} \text{cut} \frac{(V_c,V/V_c)}{w(V_C)}$$

1.2.5 Matrix Form of GWGC

We proceed by expressing $WCut(G) = \min_{V_1,V_2,....,V_k} \sum_{c=1}^{k} \frac{cut(V_c,V/V_c)}{w(V_C)}$ as a trace optimisation. With the same definitions as above the matrix D, A and L also using using the W the weighted adjacency matrix.With $w(V_C) = x_c^T W x_c$
$\min \sum_{c=1}^{k} \frac{cut(V_c,V/V_c)}{w(V_C)} = \sum_{c=1}^{k} \frac{x_c^T L x_c}{x_c^T W x_c} = \sum_{c=1}^{k} \overline{x}_c^T L \overline{x}_c$., where $\overline{x} = \frac{x_c}{(x_c^T W x_c)^{\frac{1}{2}}}$

$WCut(G) = \min \overline{x}_c^T L \overline{x}_c$, which can be expressed as a trace minimization,which is further written as $\min_Y$trace $(\overline{X}^T A \overline{X})$ $\min_Y$ trace$(Y^T W^{\frac{-1}{2}} L W^{\frac{-1}{2}} Y)$ where $Y = W^{\frac{1}{2}} \overline{X}$

1.2.6 Kernel K-Means Algorithm

K-means clustering is a method for grouping n observations into K clusters. It aims to assign each observation to the cluster with the nearest mean or centroid. The goal is to minimize the sum of squared distances between the data points and their corresponding cluster centroids, resulting in clusters that are internally homogeneous and distinct from each other. K-means is a centroid-based algorithm or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid. The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid. The goal of the optimization process is to find the best set of centroids that minimizes the sum of squared distances between each data point and its closest centroid. This process is repeated multiple times until convergence, resulting in the optimal clustering solution.

We now demonstrate the k means algorithm.

Given a given some data that we plot on a line, our objective here is to group them into 3 clusters, implying that k in this case is 3.

Figure 1.18: Data Points on a Number Line

Next we randomly select 3 different data points, these points will be our initial clusters.

We then measure the distance between the first point and the three initial cluster. Then we assign the color of the nearest cluster to the point in this example we choose the

Figure 1.19

point to be color blue, we then continue the same process with the other cluster till each one of the points has been assigned one of the three colors, according to their distance between these clusters.

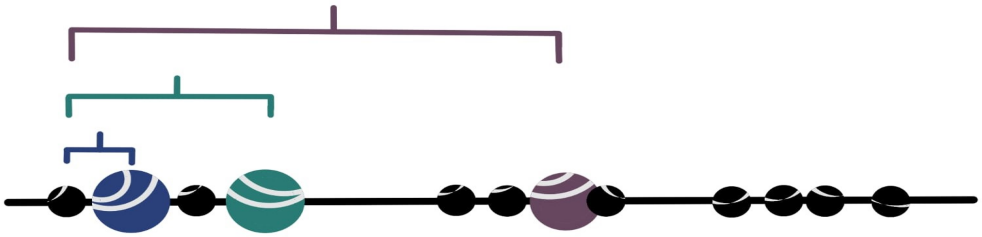


Figure 1.20

After assigning the colors ours resulting figure is as follows.

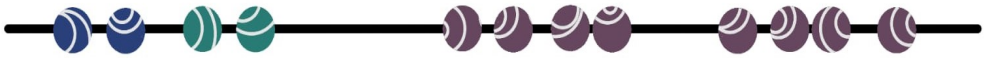


Figure 1.21

In this step we find the mean values for each cluster.

We then use the means as the cluster points and then proceed as in figure1.22, clustering the points according to their distances, and then coloring them, this is done for each step, we stop if we get the same clustering as we had got previously.

so we have obtained the same clustering as in figure1.23 hence we stop. Now we shall deal with points in a plane and how the algorithm operates there.

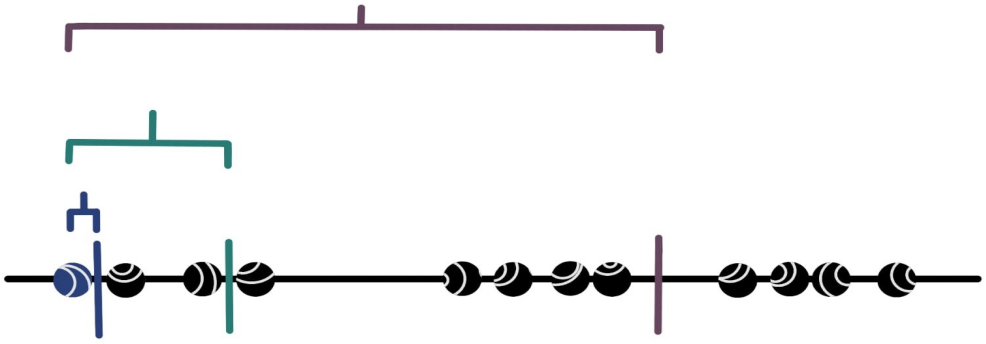


Figure 1.22

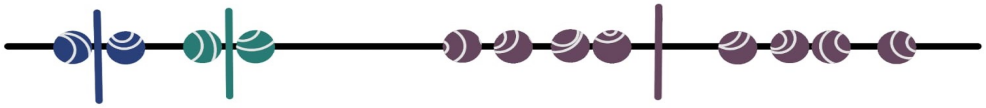


Figure 1.23

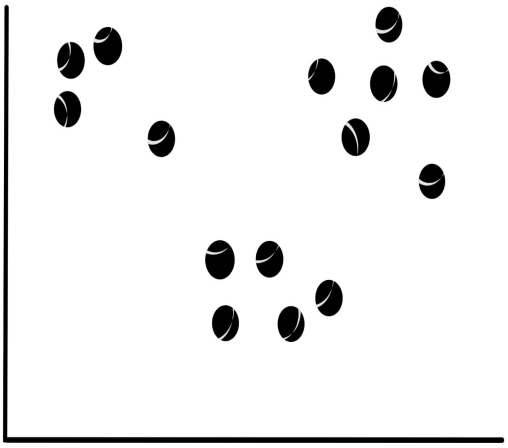


Figure 1.24

Now in a plane, given the number of clusters in our case its 3, we select 3 random points as our cluster point and then proceed by taking a point and then we find its eucledian distance between each of those cluster point, and then assign it the color of the nearest cluster.

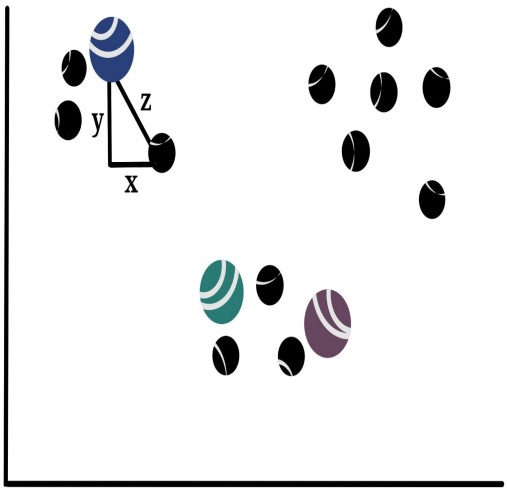


Figure 1.25

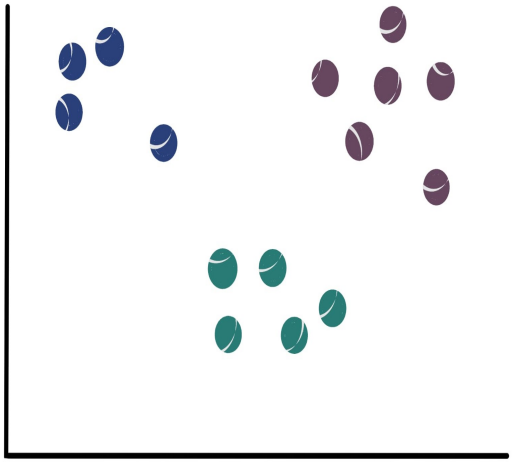


Figure 1.26

After clustering we then find the center of each cluster, and repeat the same procedure as we have done on a line.

1.2.7 The Working of Kernel k-Means Algorithm

Given a set of vectors $a_1, a_2, ..., a_n$ the k- means algorithm seeks to find partitions $\pi_1, .., \pi_k$ that minimize the objective function.

$$D(\pi_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{a_i \epsilon \pi_c} ||a_i - m_c||^2 \text{ ,where } m_c = \frac{\sum_{a_i \epsilon \pi_c a_i} a_i}{|\pi_c|}$$

Note that the c-th cluster is denoted by π_c, a clustering or a partitioning by $\pi_{c=1}^{k}$ by the centroid or the mean of the cluster is denoted by m_c.
A disadvantage of standard k-means is that clusters must be separated by a hyperplane; this follows from the fact that squared euclidean distance is used as the distortion measure. To counter this, kernel k-means issues a function to map points to a higher-dimensional feature space. When k-means is applied in this feature space, the linear separators in the feature space correspond to nonlinear separators in the input space. The kernel k-means objective can be written as a minimization of :

$$D(\pi_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{a_i \epsilon \pi_c} ||\phi(a_i) - m_c||^2 \text{ ,where } m_c = \frac{\sum_{a_i \epsilon \pi_c a_i} \phi(a_i)}{|\pi_c|}$$

If we expand the computation $||\phi(a_i) - m_c||^2$ in the objective function we can obtain the following.

$$\phi(a_i).\phi(a_i) - \frac{2\sum_{a_j \epsilon \pi_c} \phi(a_i).\phi(a_j)}{|\pi_c|} + \frac{\sum_{a_j, a_l \epsilon \pi_c} \phi(a_j).\phi(a_l)}{|\pi_c|^2}$$

Thus only inner products are used in the computation of the Euclidean distance between a point and a centroid. As a result, if we are given a kernel matrix K, where K_{ij} = $\phi(a_i).\phi(a_j)$. We can compute distances between points and centroids without knowing explicit representations of $\phi(a_i)$ and $\phi(a_j)$.

1.2.8 Weighted kernel k-means Algorithm

We now introduce a weighted version of the kernel k-means objective function. The weighted kernel k-means objective function is expressed as:

$$D(\pi_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{a_i \epsilon \pi_c} w_i ||\phi(a_i) - m_c||^2 \text{ ,where } m_c = \frac{\sum_{a_i \epsilon \pi_c a_i} w_i \phi(a_i)}{\sum_{a_i \epsilon \pi_c} w_i}$$

And the weights w_i are non-negative. Note m_c represents the best cluster representative since.

$$m_c = argmin_z \sum_{a_i \epsilon \pi_c} w_i ||\phi(a_i) - z||^2$$

As before, we compute distances only using inner products, since $||\phi(a_i) - m_c||^2$ equals

$$\phi(a_i).\phi(a_i) - \frac{2\sum_{a_j \epsilon \pi_c} w_j \phi(a_i).\phi(a_j)}{\sum_{a_j \epsilon \pi_c} w_j} + \frac{\sum_{a_j, a_l \epsilon \pi_c} w_j w_l \phi(a_j).\phi(a_l)}{(\sum_{a_j \epsilon \pi_c} w_j)^2}$$

Using the kernel matrix K, the above may be rewritten as:

$$K_{ii} - \frac{2\sum_{a_j \epsilon \pi_c} w_j K_{ij}}{\sum_{a_j \epsilon \pi_c} w_j} + \frac{\sum_{a_j, a_l \epsilon \pi_c} w_j w_l K_{jl}}{(\sum_{a_j \epsilon \pi_c} w_j)^2}$$

1.2.9 Weighted Kernel k-means as Trace Maximization

We first consider the weighted kernel k-means objective, and express it as a trace maximization problem. Let s_c be the sum of the weights in cluster c, i.e $s_c = \sum_{a_i \epsilon \pi_c} w_i$. Define the $n \times k$ matrix Z:

$$Z_{ic} = \begin{cases} \frac{1}{s_c^{\frac{1}{2}}} & \text{if } a_i \epsilon \pi_c \\ 0 & \text{otherwise} \end{cases}$$

Clearly, the columns of Z are mutually orthogonal as they capture the disjoint cluster memberships. Suppose Φ is the matrix of all $\phi(a)$ vectors and W is the diagonal matrix of the weights. It can then be verified that column i of the matrix ΦWZZ^T is equal to the mean vector of the cluster that contains a_i. Thus, the weighted kernel k-means objective may be written as:

$$D(\pi_{c=1}^k) = \sum_{c=1}^k \sum_{a_i \epsilon \pi_c} w_i ||\phi(a_i) - m_c||^2$$

$$D(\pi_{c=1}^k) = \sum_{i=1}^n w_i ||\Phi_{.}i - (\Phi WZZ^T)_{.i}||^2$$

Where $\Phi_{.i}$ denotes the i-th column of the matrix Φ. Let $\overline{Y} = W_{\frac{1}{2}}$Z; observe that $\overline{Y}$ is a orthonormal matrix ($\overline{Y}^T \overline{Y} = I_k$). Then we write the objective function as :

$$\begin{aligned} D(\pi_{c=1}^k) &= \sum_{i=1}^n w_i ||\Phi_{.i} - (\Phi W^{\frac{1}{2}} \overline{Y}\overline{Y}^T W^{\frac{-1}{2}})_{.i}||^2. \\ &= \sum_{i=1}^n ||\Phi_{.i} w_i^{\frac{1}{2}} - (\Phi W^{\frac{1}{2}} \overline{Y}\overline{Y}^T)_{.i}||^2 \\ &= ||\Phi_{.i} W^{\frac{1}{2}} - (\Phi W^{\frac{1}{2}} \overline{Y}\overline{Y}^T)_{.i}||^2 \end{aligned}$$

Using the fact that trace(AA^T) = trace($A^T A$) = $||A||_F^2$, trace(A+B) = trace(A)+trace(B) and trace(AB) = trace(BA) we have :

$$D(\pi_{c=1}^k) = \text{trace}(W^{\frac{1}{2}}\Phi^T\Phi W^{\frac{1}{2}} - W^{\frac{1}{2}}\Phi^T\Phi W^{\frac{1}{2}}\overline{Y}\overline{Y}^T - \overline{Y}\overline{Y}^T W^{\frac{1}{2}}\Phi^T\Phi W^{\frac{1}{2}} + \overline{Y}\overline{Y}^T W^{\frac{1}{2}}\Phi^T\Phi W^{\frac{1}{2}}\overline{Y}\overline{Y}^T)$$

= trace($W^{\frac{1}{2}}\Phi^T\Phi W^{\frac{1}{2}}$) - trace($\overline{Y}^T W^{\frac{1}{2}}\Phi^T\Phi W^{\frac{1}{2}}\overline{Y}$)

We note that the kernal matrix K is equal to $\Phi^T\Phi$ and that trace($W^{\frac{1}{2}}KW^{\frac{1}{2}}$) is a constant. Therefore the minimization of the weighted kernel k-means objective function is equivalent to:

$$max_{\overline{Y}}\text{trace}(\overline{Y}^T W^{\frac{1}{2}}\Phi^T\Phi W^{\frac{1}{2}}\overline{Y})$$

Where $\overline{Y}$ is an orthonormal $n \times k$ matrix that is proportional to the square root of the weight matrix W.

Chapter 2

DAC METHOD

2.1 Construction of The Weighted Graph

For a given airspace, we assume that the static structure information includes air routes and key points such as airports, waypoints and crossing points is known in advance. According to the structure information, we set up an undirected graph G, we have a graph $G = G(V, E)$, where the vertex set $V = \{1, 2, ..., n\}$ consists of the key points. The edge set $E = \{(ij) : i, j \epsilon \mathrm{V}\}$ represents the air routes. Aircraft count will be adopted as the workload metric. Both vertices and edges in the undirected graph can be assigned with those aircraft counts, and hence forming our graph model.

For the vertices and edges of the graph being assigned with weights, a Voronoi diagram D is built as in figure 2.1, whose sites(points) are the vertices of the undirected graph. D decomposes by its borders the airspace into a series of units called as Voronoi cells(regions), C_i $(i = 1, 2, ..., n)$.

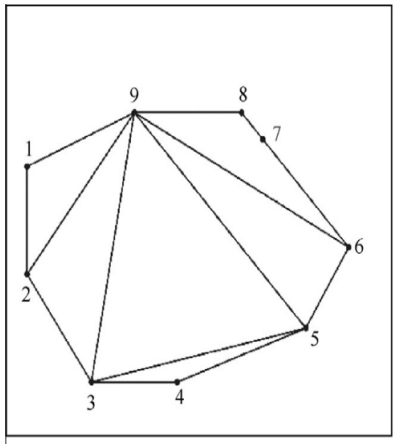


Figure 2.1

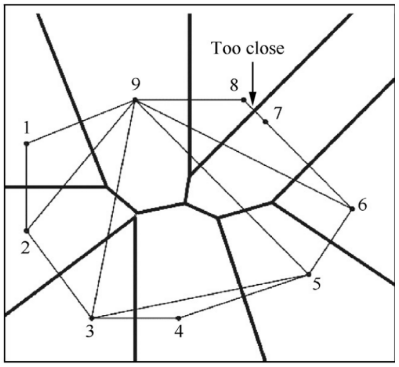


Figure 2.2

As one can see in figure 2.2 that each cell corresponds only to one site. Thick lines represent borders of cells, and thin lines represent air routes. In the above fig2.2, 1,2,..., 9 represents vertex index (1),(2),..., (8) represents the vertex weight[1],[2],...,[8] represents the edge weight. We can see that there may be a case that some of the sites or the air routes are close to the cell borders. This leads to a result that some of the designed sectors will not satisfy the minimum distance constraint. If the sector boundaries coincide with those borders, so the borders have to be removed. The cells that are adjacent to those borders are combined into new cells.

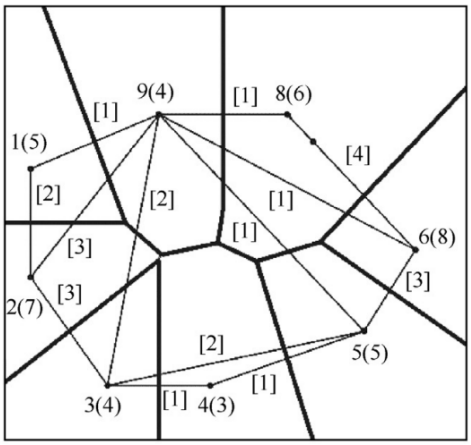


Figure 2.3

The vertex v_i represents the cell C_i. The weight on the vertex v_i represents the aircraft count w_i in the corresponding cell C_i. The edge e_{ij} represents all the air routes between cells C_i and C_j. The edge weight w_{ij} describes the sum of aircraft counts along all the air routes between the cells C_i and C_j. For G_w all aircraft counts on vertices are represented as a vector w. All aircraft counts along the edges among the cells are described by a matrix W .

when the airspace is described as G_w, the DAC problem with the objective of balancing sector workloads, minimizing the coordination workload, this problems are then converted into graph partitioning problems of maximizing the sub-graph weight balance, minimizing the edge weight among the sub-graphs $G^{i'}$ $(i = 1, 2, ...K)$.

The DAC objective can be described mathematically as a graph partitioning objective by the following functions. $\min_{G_w^{1'}, G_w^{2'}, ..., G_w^{k'}} \sum_{c=1}^{k} \frac{cut(G_W^{i'}, G_W/G_W^{i'})}{w(G_W^{i'})}$ Subject to $w(G_W^{i'}) = w(G_W^{j'})$,

$$w(G_W^{i'}) = \sum_{V_c \in G_W^{i'}} w_c$$
$$\text{cut}(G_W^{i'}, G_W/G_W^{i'}) = \sum_{V_c \in G_W^{i'}, V_d \notin G_W^{i'}} w_{cd}$$

Here, $w(G_W^{i'})$ is the weight of the ith sub graph. is the no of sectors which can be determined by total aircraft count of the given airspace A_{count} and the maximum aircraft count of a sector S_{count} as follows. $k = \lceil \frac{A_{count}}{S_{count}} \rceil$.

2.2 Partition of The Weighted Graph Model

After obtaining our graph from the given data we name the graph as $G_w = \{V_w, E_w, w, W\}$ where $V_w = v_1, v_2, .., v_n$ is a vertex set and $E_w = \{e_{ij} : v_i, v_j \in V_w\}$ is an edge set in which e_{ij} is the edge connecting v_i and v_j.

Our graph being a weighted graph, with the vertices and edges having weights where the weights represent the air craft count during a specified time period.

Each vertex weight is given by w_i (i=1,..,r) where $r \leq n$, n is the no. of vertices in the graph G_w.The edge weights are given by w_{ij} . now all the vertex weights(aircraft counts) are represented by a vector w = $[w_1, w_2, ..., w_r]^T$, and the W =$[w_{ij}]_{r \times r}$, $w_{ij} = w_{ji}$ is the matrix describing the aircraft counts along all the edges, the matrix W is a adjacency matrix with values being the edge weights or a weighted adjacency matrix.

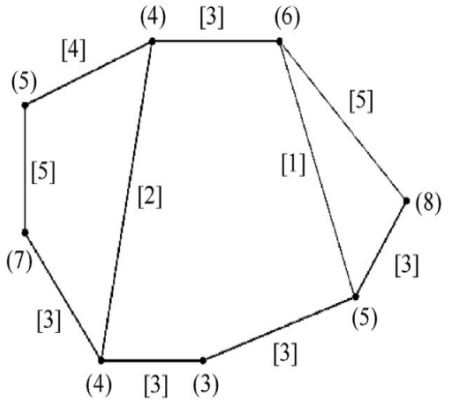


Figure 2.4

The given figure2.4 represents a weighted graph as defined above, now here the vector w is given by w = $[5,7,4,3,5,8,6,4]^T$.
The matrix W is as follows,where we start with vertex weight 5 and move anticlockwise where the next vertex has weight 7 and so forth.

$$\mathrm{W} = \begin{Vmatrix} 0 & 5 & 0 & 0 & 0 & 0 & 0 & 4 \\ 5 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 3 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 1 & 5 & 0 & 3 \\ 4 & 0 & 2 & 0 & 0 & 0 & 3 & 0 \end{Vmatrix}$$

2.2.1 Partitioning G_w by GWGC Algorithm

GWGC- general weighted graph cuts algorithm. In this section we will use the GWGC algorithm to partition the graph G_w into k subgraphs $G_w^{i'}$ $(i = 1,2,...,k)$. The algorithm can achieve an objective an objective defined as $J = \min_{G_w^1, G_w^2, ..., G_w^k} \sum_{i=1}^{k} \mathrm{cut} \frac{(G_W^i, G_W / G_W^i)}{w(G_W^i)}$ this is general weighted graph cuts problem and its solution is designed as follows.

Step 1. We create a diagonal matrix T whose entries are obtained by summing all entries in the corresponding column of the matrix W. We compute the Laplacian matrix $L = T - W$.
Step 2. $s_i = \frac{1}{\sqrt{w_i}}$ $(i = 1,2,3,...,r)$, $S = \mathrm{diag}(s_1, s_2, s_3, ..., s_r)$ and calculate $C = S \times L \times S$.
Step 3. Calculate the eigen values of C, assume that they are $\lambda_1 = 0 \leq \lambda_2 \leq, ..., \leq \lambda_r$ in ascending order.
Then, use k-means algorithm to cluster k vectors corresponding to eigenvalues from λ_1

to λ_k . From k clusters we can get a series G_w^i $(i = 1,2,...,k)$ that holds connectivity. When G_w is partitioned into a series of sub-graphs by GWGC algorithm, the sub-graphs meet the property which follows the objective.

2.3 Calculating the Vertex Weights to be Transferred by ODLB Algorithm

After partitioning the graph G_w into k subgraphs we move onto achieve the following objective given by $w(G_w^i) = w(G_w^j)$, i.e the sum of the vertex weights in each subgraph should be made equal, to accomplish this task we need to transfer vertex weights, between subgraphs meaning we will be changing the aircraft counts in each subgraph.
We will now implement the ODLB (optimal dynamic load balancing) algorithm to balance the vertex weights in each subgraph that we have obtained. The ODLB algorithm gives the direction in which the vertex is moved as well as the amount of the vertex weight to be transferred.
We proceed further by showing an example.
Provided that by means of GWGC algorithm G_w is partitioned into four subgraphs $G_w^1, G_w^2, G_w^3, G_w^4$ and accordingly the weights of the four subgraphs will be given.
Our objective is to achieve $w(G_w^i) = w(G_w^j)$ i.e the sum of the vertex weights in each subgraph must be made equal.
In order to achieve the objective above we transfer weights denoted as x_{ij} from $w(G_w^i)$ to $w(G_w^j)$, along the cut edges between the subgraphs, if subgraphs do not have cut edges in between them then vertex weights cannot be transferred among them.
Next the average weights per sub graph is as follows, $\overline{w} = \frac{w(G_w^1)+w(G_w^2)+w(G_w^3)+w(G_w^4)}{4}$.
Which nothing but the mean of the vertex weights of the 4 subgraphs.
We suppose that the weight will be shifted along the direction from the sub-graph with

small index to the sub-graph with large index. The former is called the head of the direction and the latter is called the tail.

So, F is defined as. $F_{ij} = \begin{cases} 1 & \text{if sub graph i is the head of the direction of shift} \\ -1 & \text{if sub graph j is the tail of the direction of shift} \\ 0 & \text{otherwise} \end{cases}$

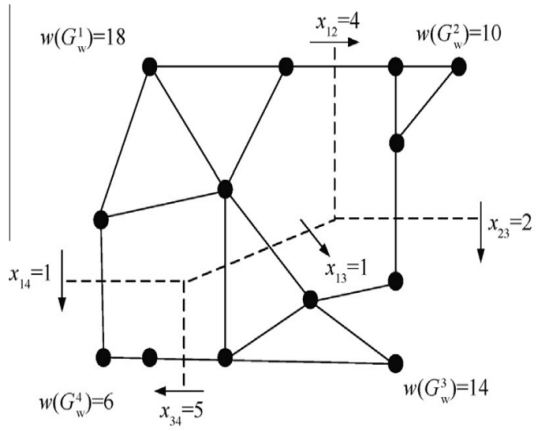


Figure 2.5

We get the following equations,

$$x_{12} + x_{13} + x_{14} = w(G^1_w) - \overline{w}$$

$$-x_{12} + x_{23} = w(G^2_w) - \overline{w}$$

$$-x_{13} - x_{23} + x_{34} = w(G^3_w) - \overline{w}$$

$$-x_{14} - x_{34} = w(G^4_w) - \overline{w}$$

Furthermore the equation can be described as $Fx = b$,where $b = [w(G_w^1) - \overline{w}, w(G_w^2) - \overline{w}, w(G_w^3) - \overline{w}, w(G_w^4) - \overline{w}]^T$, $x = [x_{12}, x_{13}, x_{14}, x_{23}, x_{34}]^T$ and $F = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 & 1 \\ 0 & 0 & -1 & 0 & -1 \end{bmatrix}$

$w(G_w^1) = 18$, $w(G_w^2) = 10$, $w(G_w^3) = 14$, $w(G_w^4) = 6$

$\overline{w} = \frac{w(G_w^1)+w(G_w^2)+w(G_w^3)+w(G_w^4)}{4}$

$\overline{w} = 12$

$x_{12} + x_{13} + x_{14} = 6$

$-x_{12} + x_{23} = -2$

$-x_{13} - x_{23} + x_{34} = 2$

$-x_{14} - x_{34} = -6$

Since our example is on a smaller scale we can find the solutions manually keeping in mind that sum of all vertex weights of subgraphs should be $\overline{w} = 12$, hence transfers are made.

This is a system of linear inhomogeneous equations to be solved, where there are five variables and four equations. The knowledge of linear algebra verifies that there are infinite solutions to the equations. Among solutions, the solution for the migration of less weight is preferred.

To solve such a problem we follow the procedure.

given $Fx = b$,

Step 1. Calculate $\overline{w} = \frac{1}{k} \sum_{i=1}^{k} w(G_w^i)$, and $b = [w(G_w^1) - \overline{w}, w(G_w^2) - \overline{w}, ..., w(G_w^k) - \overline{w}]^T$.

Step 2. Construct the matrix of subgraphs F, and calculate matrix $L = FF^T$. $x = F^T d$ where is x vector. Then by substituting x in $Fx = b$, we get $FF^T d = b$.

Step 3. Construct a linear equation system $Ld = b$ and solve it for d. One can conclude that from the sub-graph i to j is obtained by $F^T d$. New weights transfers obtained from the above method are given by.

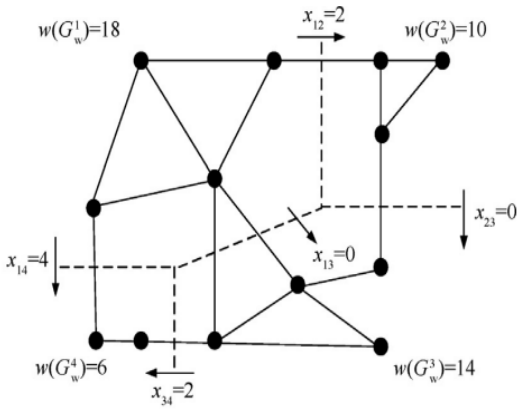


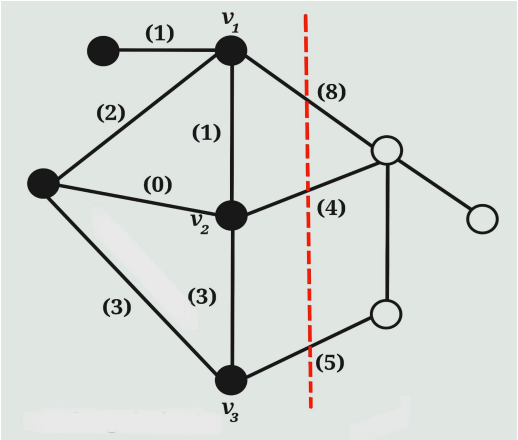
Figure 2.6

From the ODLB algorithm, we know a fact thatx_{ij} may be a negative value or positive value. The positive value means the weights move from G_w^i to G_w^j while a negative value means the weights move from G_w^j to G_w^i. In addition, x_{ij} may not be the integers, but decimals. However, x_{ij} is the aircraft count, this means x_{ij} is integer. So the measure must be taken to deal with the problem by rounding x_{ij} to be a whole number.

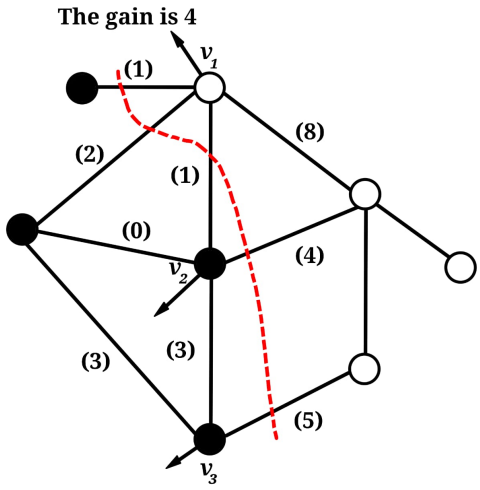
2.4 Transferring the Vertices

Using the ODLB algorithm we found x_{ij} (weight to be transferred among two subgraphs). So that the sum of the vertex weights in each subgraph can be made equal. Now we provide the vertices in each subrgaph that need to be transferred.

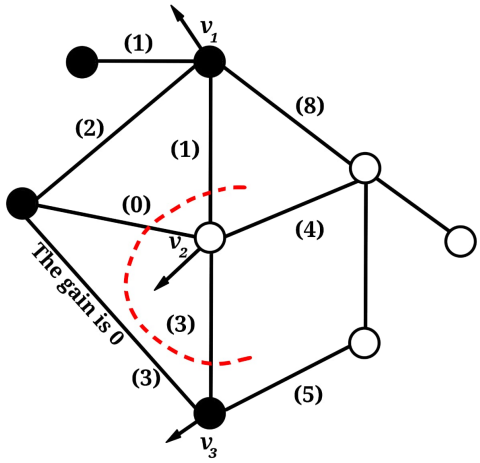
The two subgraphs in the diagram are separated by a dotted line. In this example are subgraph the sum of the vertex weight of each subgraph is not balanced. We intend to transfer five weights from the left sub-graph to the right. When both weights of v_1 and v_2 are 5, which vertex should be transferred, v_1 or v_2 ?.



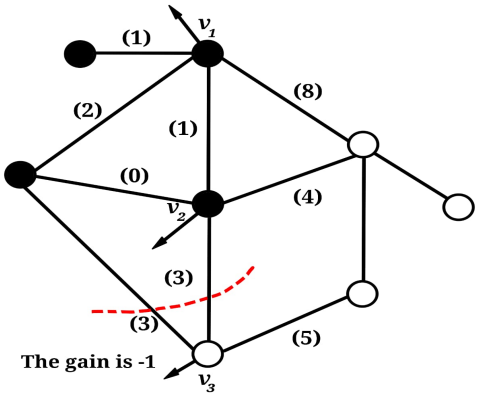
So we begin by selecting one of the two vertices in no particular order. When v_1 is moved to the right subgraph, the edge weight connecting two sub-graphs will decrease. The change in the edge weight can be gotten from the following equation: 8-(1+2+1) = 4, this means that coordination workload (cut edge weight) will be reduced. Since after transferring the vertex the subgraph changes and so do the cut edges.



similarly when v_2 is moved the edge weight does not change anymore. As the gain here can be given by 4-4 = 0



Now we check for v_3, here the gain being 5-6 = -1, this is the worst vertex to be transferred as it increases the gain by 1.



The change in the edge weight for the vertex migration is called the gain. The equation is generalized to calculate the gain g_d for the vertex v_d. $g_d = \sum_{v_d \epsilon G_w^i, v_f \epsilon G_w^j} w_{df}$ -$\sum_{v_d, v_e \epsilon G_w^i,} w_{de}$.

The algorithm is as follows. firstly, according to x_{ij}, we can determine the migration direction of the vertices. Let the set of vertices in G_w^i adjacent to G_w^j adjacent be denoted as B_{ij}. The sum of the weights corresponding to the vertices in B_{ij} be a_{ij}.the gain g_d of v_d in B_{ij} be determined by $g_d = \sum_{v_d \epsilon G_w^i, v_f \epsilon G_w^j} w_{df}$ -$\sum_{v_d, v_e \epsilon G_w^i,} w_{de}$. The vertices in B_{ij} are sorted according to their gains by a descending order. The vertex in B_{ij} the vertex with the largest gains is transferred to G_w^j, and the procedure is repeated according to the descending order. If $a_{ij} < x_{ij}$, after transferring all the vertices in B_{ij}, the procedure above can continue until the required x_{ij} has been satisfied for new vertices in G_w^i adjacent to G_w^j will appear after migrating all vertices in B_{ij}. In this we can get a series of new subgraphs $G_w^{i'}$ $(i = 1, 2, ...k)$ from G_w^i with equalized $w(G_w^{i'})$.
The new subgraphs $G_w^{i'}$ $(i = 1, 2, ...k)$ satisfy the properties as follows. $w(G_w^{i'})$ =$w(G_w^{j'})$, $G_w = \bigcup_{i=1}^{k} G_w^{i'}$, $G_w^{i'} \cap G_W^{j'} = \varnothing$.**??**

Chapter 3

ANALYSIS AND CONCLUSION

The paper has presented a new method for DAC based on a weighted graph model by applying GWGC algorithm and ODLB algorithm in combination with a heuristic algorithm inspired from the gain of K-L algorithm to partition given airspace into sectors achieving the objective of balancing the workloads and of minimizing the coordination workloads among the designed sectors.

1. The designed sectors have balanced aircraft count while coordination workload is minimized
2. The designed sectors satisfy geometrical constraints, such as convexity constraint, connectivity constraint and minimum distance constraint.
3. And more importantly, the low traffic results in fewer sectors as compared to the current airspace configuration. The consequence being reduction in the number of controllers, and thereby the administrative expense reduced.

Bibliography

[1] Yangzhou Chen and Defu Zhang. "Dynamic airspace configuration method based on a weighted graph model". In: *Chinese Journal of Aeronautics* 27 (4 2014), pp. 903–912. ISSN: 1000-9361. DOI: https://doi.org/10.1016/j.cja.2014.06.009. URL: https://www.sciencedirect.com/science/article/pii/S1000936114001174.

[2] Zhang Defu et al. "Airspace Sectorization Based on Weighted Graph Spectral Bisection Algorithm". In: Apr. 2012, pp. 1924–1935. ISBN: 978-0-7844-1244-2. DOI: 10.1061/9780784412442.196.

[3] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. "Weighted Graph Cuts without Eigenvectors A Multilevel Approach". In: *IEEE transactions on pattern analysis and machine intelligence* 29 (Apr. 2007), pp. 1944–1957. DOI: 10.1109/TPAMI.2007.1115.

[4] Stephane Martinez et al. "A Weighted-Graph Approach for Dynamic Airspace Configuration". In: (Feb. 2007). DOI: 10.2514/6.2007-6448.

[5] Roberto Tamassia and Scott S. Snibbe. *Introduction to Voronoi diagrams*. Feb. 1993. URL: https://cs.brown.edu/courses/cs252/misc/resources/lectures/pdf/notes09.pdf.

[6] Huy Trandac, Philippe Baptiste, and Vu Duong. "Airspace sectorization with constraints". In: *RAIRO - Operations Research* 39 (Feb. 2005), pp. 105–122. DOI: 10.1051/ro:2005005.

[7] Min Xue. “Airspace Sector Redesign Based on Voronoi Diagrams”. In: *J. Aerosp. Comput. Inf. Commun.* 6 (2008), pp. 624–634. URL: https://api.semanticscholar.org/CorpusID:6360114.