Understanding Atmospheric Chemistry using Graph-Theory, Visualisation and Machine Learning.

Dan Ellis

March 2020

Veritatem inquirenti, semel in vita de omnibus, quantum fieri potest, esse dubitandu:
In order to seek truth, it is necessary once in the course of our life, to doubt, as far as possible, of all things.

- Descartes, Rene, *Principles of Philosophy*

Contents

Chapter 1

Applying Visual Analytics to the Atmospheric Chemistry Network

“ I have a notion that when the mind is thinking, it is simply talking to itself, asking questions and answering them. ”

- Socrates, *The collected dialogues of Plato*

1.1 Introduction

?? viewed the importance of a carefully selected visualisation/metaphor in the representation of scientific data. One such category is that of relational data, where we have a set of items, joined by a chosen relationship. Historically this type of problem was solved using sociographs to present a set of items and the links between them.

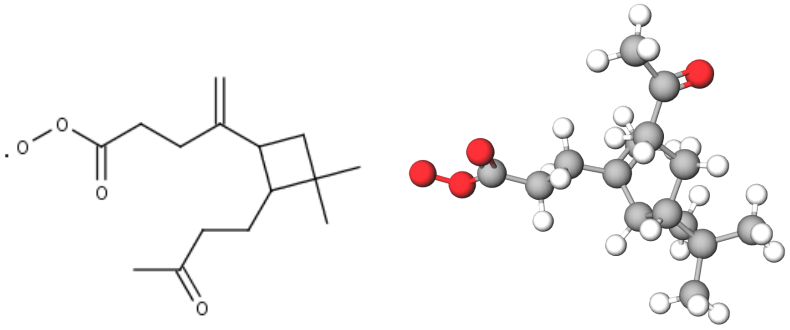
This chapter begins by looking at the use of sociograms in chemistry (Subsection 1.1.2) and the different ways in which these can help convey information to the reader (Section 1.2, Section 1.4). These sections find the force-directed graph to be the most suited for representing the chemical reactions within a mechanism, and therefore this shall be applied to the network of reactions representing the chemistry within an urban environment - Beijing (Section 1.5).

1.1.1 Networks And Their Role In Visual Analytics

Networks are present everywhere - this ranges from interactions within social media to bank transactions, internet routing, genetics to epidemiology [Martin Grandjean, 2016; Staples et al., 2013; Needham and Hodler, 2019; Baronchelli et al., 2013; Sangers et al., 2019; Kohlbacher et al., 2014; Archambault et al., 2014; Schreiber et al., 2014]. The sociogram (or graph) structure can be applied to any set of items which contain one or more relationships between them. In visualisation, these 'items' are referred to as nodes/vertices, and their relationships as edges/links [Kerren et al., 2014] - terms that will be used interchangeably throughout this thesis.

1.1.2 Graphs In Chemistry

Node-link representations have been at the core of chemistry for many years. They have been used to show the bonds between atoms and are integral to the representation of molecules - both physically (with the aid of molecular model kits) or pictorially to show various structural properties (Figure 1.1). These graph-like analogies provide a pseudo-physical representation of the molecules and their reactions in a way that is intuitive to the user. Subsubsection 1.1.2.1 shows the use of a sociograph structure to represent reactions within the troposphere; however, this method of representation is not limited to atmospheric science - for example, Figure 1.3 depicts the biochemical metabolic pathways of the human body. This is an example of another complex chemical network that benefits from this method of representation.



(a) 2D 'classic' (b) 3D

Figure 1.1: **The molecule $C_{141}CO_3$ shown in both 2D and 3D node-link structures.** This is a the result of a series of inorganic species reactions and a desocciation from BCARY - the only sesqueterpine in the MCM. 3D visualisation by [Bergwerf, 2019].

1.1.2.1 Using Sociograms To Describe Reactions

A collection of reactions representing the chemistry of a region is called a mechanism. The Master Chemical Mechanism [**?**] provides a collection of equations describing the gas-phase chemistry which exists within the troposphere (**??**). In its use in policy, and the evaluation of Air Quality Models ([Dick Derwent, 2010]), it is often useful to understand the degradation process different VOCs undergo. In general, this can be achieved through a series of interconnected reactions in the form of a reaction cycle (Figure 1.2). This type of sociograph shows the directional nature of chemical reactions and the relationships between different species. This has many similarities to a conventional directed graph, except that species (nodes) are sometimes duplicated (for example OH, HO_2, O_2 in Figure 1.2) to aid in the clarity of the figure.

This provides an excellent example of how the flow-like nature of a sociogram aids in the understanding of a potentially complex chemical system of 171 organic species and 600 reactions. Evolutionary traits, including the genetic predisposition to interpret shapes faster than text ([Harari, 2015]) make the graph structure a much better method for representing such a system.

1.1.3 Modeling Chemistry As A Directed Graph.

Historically it is shown that the graph format has proven to be an efficient means of understanding the reactions within a mechanism. Traditionally these are constructed manually, with the designer making a series of choices on how best to place, and simplify the chemistry based on their application. As our understanding of chemistry improves and we have started to progress into automated and semi-automated mechanism construction. This makes the construction of mechanisms with tens of millions of species and billions of reaction possible ([Aumont et al., 2005]) and is the point where the manual design/simplification of reaction networks becomes infeasible.

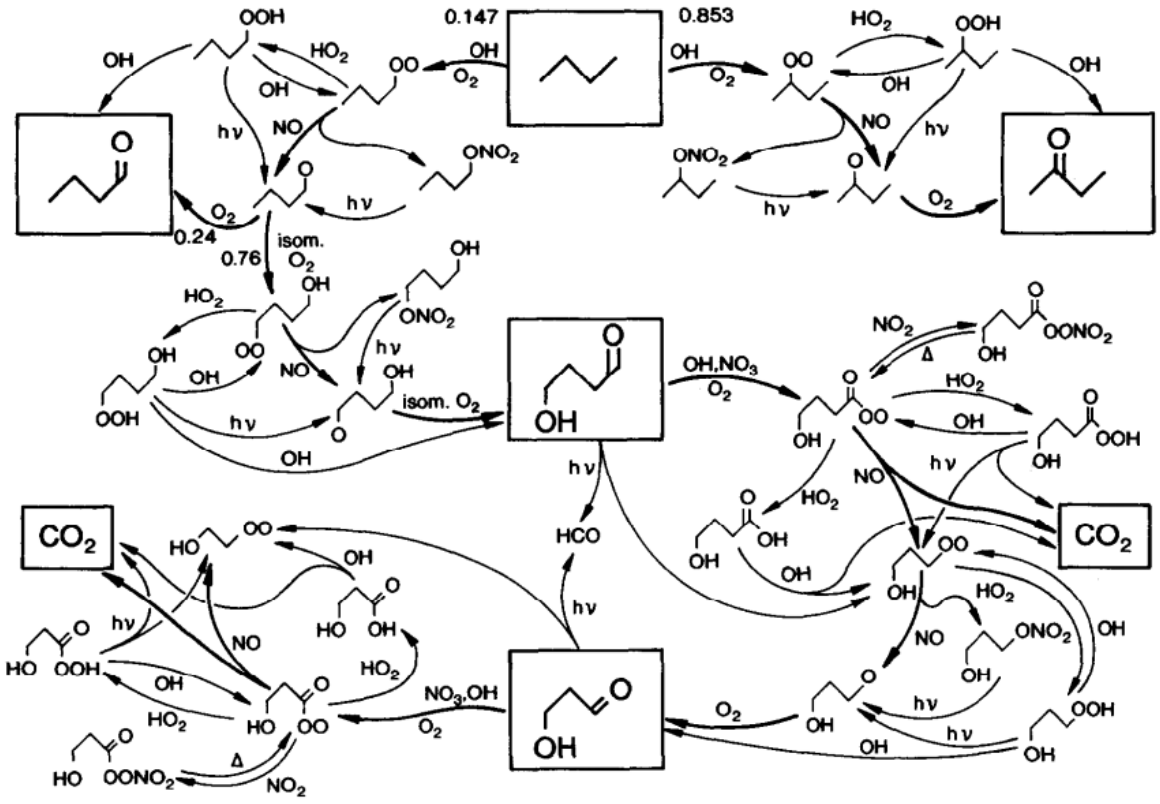


Figure 1.2: **A systematic representation of the degregation of butane.** Using this we are able to see the process C_4H_{10} undergoes before its ultimate demise as carbon monoxide and water. Source: [Jenkin et al., 1997]

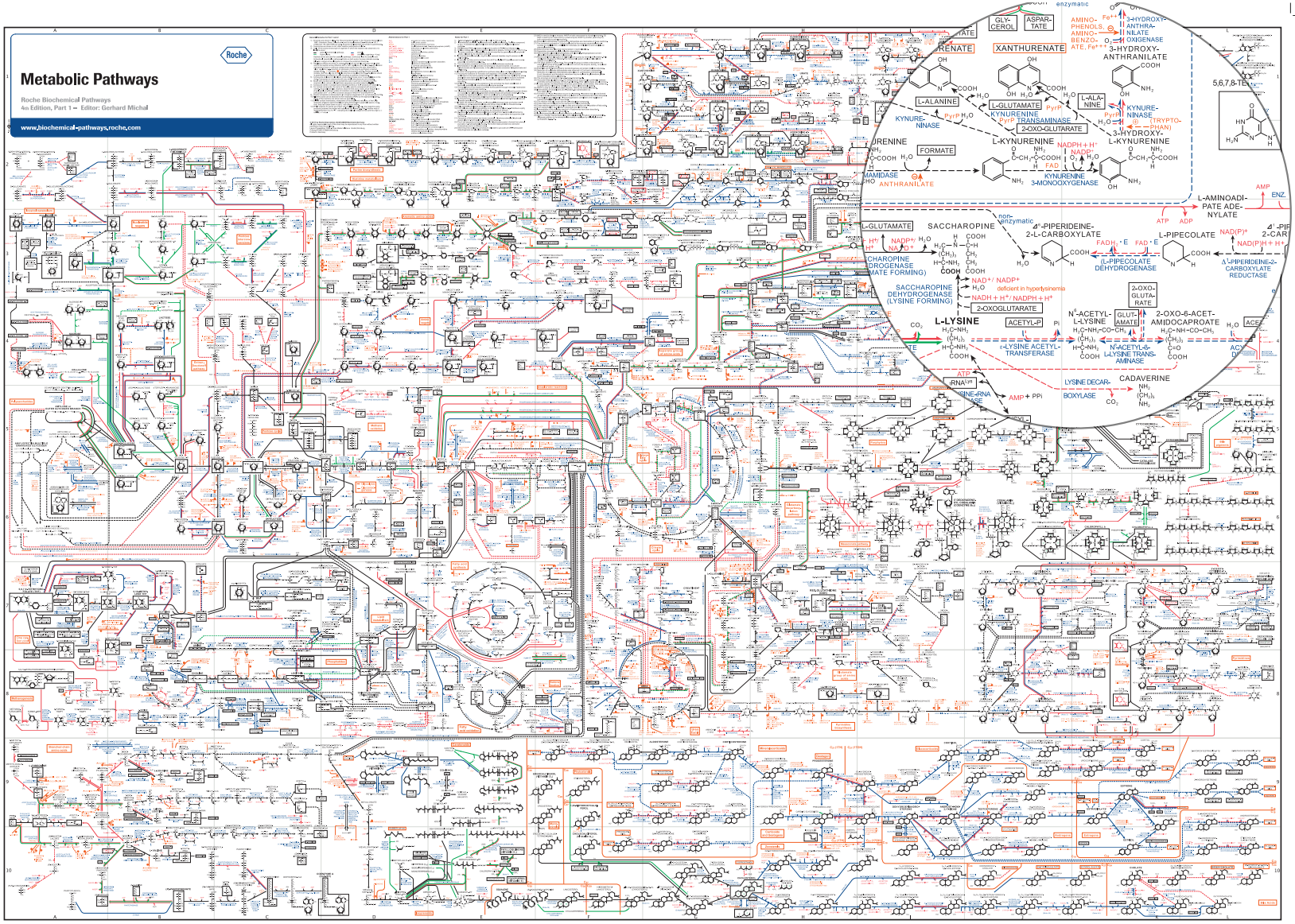


Figure 1.3: **The Roche Metabolic Pathways of the human body.** This example demonstrates the ability to manually represent the complex chemistry of the body using a graph structure. (Original A0 version is available at the source). Source: [Michal, 1965]

Today automatic graph layouts allow us to generate multivariate and complex graphs quickly [Muelder et al., 2014] -This means that, much like in the construction of a mechanism, we can rely on computer-aided design to generate a directed graph representation of the chemistry. Montañez [2016] states that "The beauty of a good information graphic is that it can tell a whole story in a single unit of visual content". This is particularly true for the use of directed graphs in chemistry where we can compare different mechanism subsets,(**??**) or model simulations (**??**).

However, several problems emerge from the complete automation of a task. Firstly real-world data very rarely reacts how it is expected to. Here networks of high edge density often obfuscate the graph data and produce what is only described as a 'birds nest', 'hairball' or 'ball of yarn' within the literature [Roberts et al., 2014]. Although such problems can be shown as moments of turbulence, they encourage a greater understanding of the graphic design process and can catalyze to merge unique ideas into an effective visualisation [Johnson, 2010] - much like the composite metaphors in **??**.

Having established that a graph network ties in both modern and historical methods for representing relational data, we now look at how to present the graph, both in syntax (Section 1.2) and semantics (Section 1.4).

1.2 Graph Syntatics

Syntactic representation considers how best to distribute information on a page for maximum impact. This can be seen between the force-directed graph (top) and geographical location (bottom) layouts in Figure 1.4. Although the geographical layout gives a more accurate representation of the distances between unconnected nodes (airports), a force-directed graph provides greater insight into the relationships (flights) between each airport. This highlights the importance of choosing a suitable syntactic representation to highlight the features of interest. The remainder of this section discusses the syntactic choices required for the visualisation of a complex chemical mechanism.

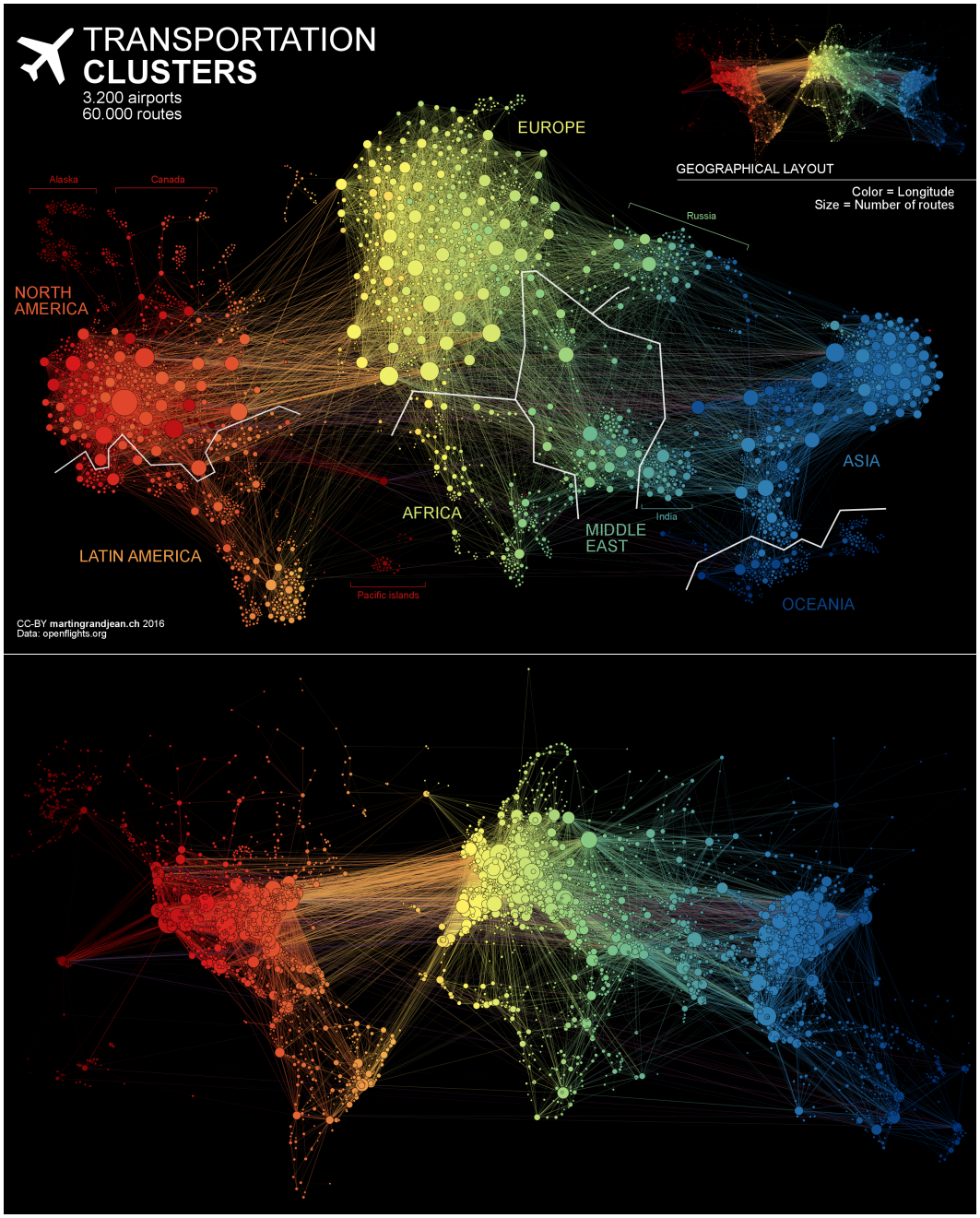


Figure 1.4: **Comparison of different representations of flight data by [Martin Grandjean, 2016].** The top figure shows the data represented by a force-directed graph layout (described below) and a Geo-layout showing each point at its location on the Earth.

1.2.1 Selecting The Correct Evaluation Criteria.

As chemical networks provide a wealth of information on the reactions within a system, this can prove challenging to user cognition and computational resources [Kerren et al., 2014]. In selecting the best possible graph layout, there are many metrics designed around the improving of visualisations aesthetics [Purchase, 2002]; however, these have often only been evaluated with a handful of criteria in mind. Such metrics can make it difficult to accurately quantify the changes in user-readability,

especially if they are not treated as originally intended [Pohl et al., 2009].

1.2.1.1 Edge Crossing

One of the greatest limitations to understanding a graph is the number of overlapping (crossing) edges [Purchase, 1997], especially since users often spend most of their time looking at the edges of a graph in order to understand it [Pohl et al., 2009].

There exist several types of graph layout algorithms which aim to reduce the number of overlapping edges in a graph. The two most common ones are force-directed and orthogonal. Orthogonal designs are those of straight edges at 90-degree angles, such as in architectural or circuit schematics (Figure 1.5). Force-directed graphs (Subsubsection 1.2.2.3) are a graph layout is designed to simulate a physical system, where node positions are the result of the push and pull of the edges between them. In the task selecting nodes from a specific path, users were twice as more accurate using this layout than the orthogonal one [Pohl et al., 2009].

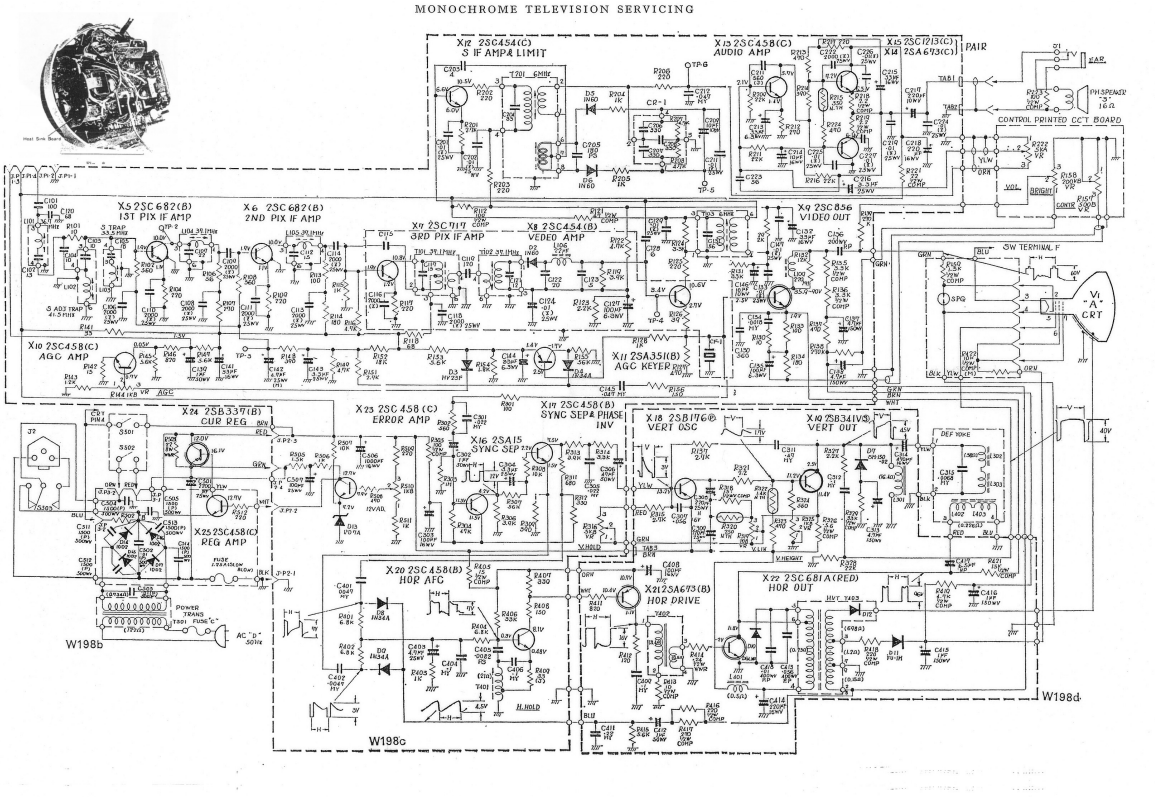


Figure 1.5: **An orthogonal circuit schematic of the Model 3 240 portable cathode ray tube television.** The circuit schematic of the television (top left) shows a much simpler representation of how different components within the television are connected. Source: [**?**]

1.2.1.2 Node Distribution And Overlap

The distribution of nodes across the page can both hinder or increase the readability of a graph - especially since larger nodes may obscure smaller ones at the same location. Purchase et al. [2003] found that graphs with an equal node distribution across space, at a medium edge length greatly improved graph readability - with node distribution and graph-symmetry ranking second in a study on user preference on graphs.

In addition to selecting the best graph layout, there exist several methods in which overlapping nodes may be removed - an issue that is sometimes difficult by the treating of nodes as 'point masses' within an algorithm [Dwyer et al., 2006c]. Dwyer et al. [2006b] explains that there are usually two methods for reducing the number of overlapping nodes in a graph; these are:

1. Create a layout design capable of taking node size (e.g. [Friedrich and Schreiber, 2004]) into consideration. These designs tend to be layout specific and not absolute in removing all overlap between nodes.

2. This requires a level of post-processing in the form of a 'layout adjustment'. Here we reposition nodes after a chosen layout has finished computing. The drawback of this method is that information contained in the graph's shape may be degraded. This can be done through the use of collision detection, or moving nodes to the centre of the vernouli cells [Lyons, 1992].

1.2.2 Automated Graph Drawing Layouts

In their design and evaluation, automatic graph drawing algorithms are created to minimise a specific criterion. This subsection will compare several graph layouts and make a verdict on which one is most suited for the representation of tropospheric chemistry. This task shall use the mechanism extracted in **??** to represent the VOC's within the Beijing city - a real-world case study using the MCM.

To do this we begin by exploring hand-drawn / map inspired graph layouts (Subsubsection 1.2.2.1,Subsubsection 1
eventualy ending at a number of automated force-directed graphs (Subsubsection 1.2.2.3).

1.2.2.1 Replication Of Hand-Drawing Methods

With the rise of computation, many traditional visualisations adapted for the computer-aided generation. Fields of architecture and circuit design adopted computational software to alleviate some of the difficulties presented by large or complex designs. Similar ideas such as the use of automatically generated transit maps can be used to link chronological or topological items such as ideas [Foo, 2019]. Figure 1.6 shows all the possible paths for the oxidation of methane to produce carbon dioxide

(and water), using the MemoryMap algorithm Foo [2019]. Although such methods can be useful in showing isolated pathways, they provide a convoluted representation of large interconnected systems and require some manual intervention.

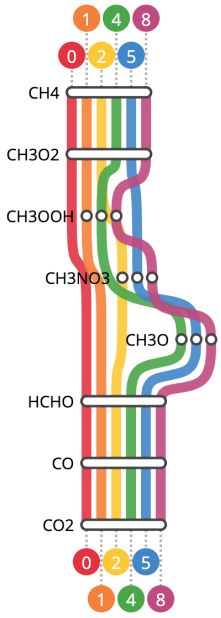
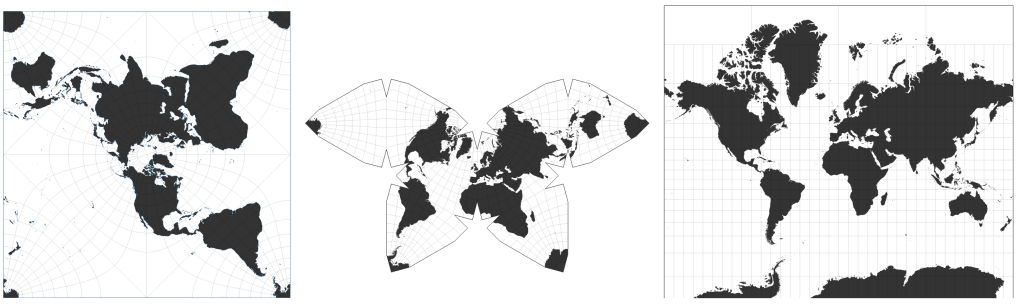


Figure 1.6: **A transit map showing all the possible routes from methane to carbon dioxide.** This was drawn using MemoryMap [Foo, 2019] and uses a version of the MCM methane subset, where carbon dioxide has been introduced.

1.2.2.2 Projection Based

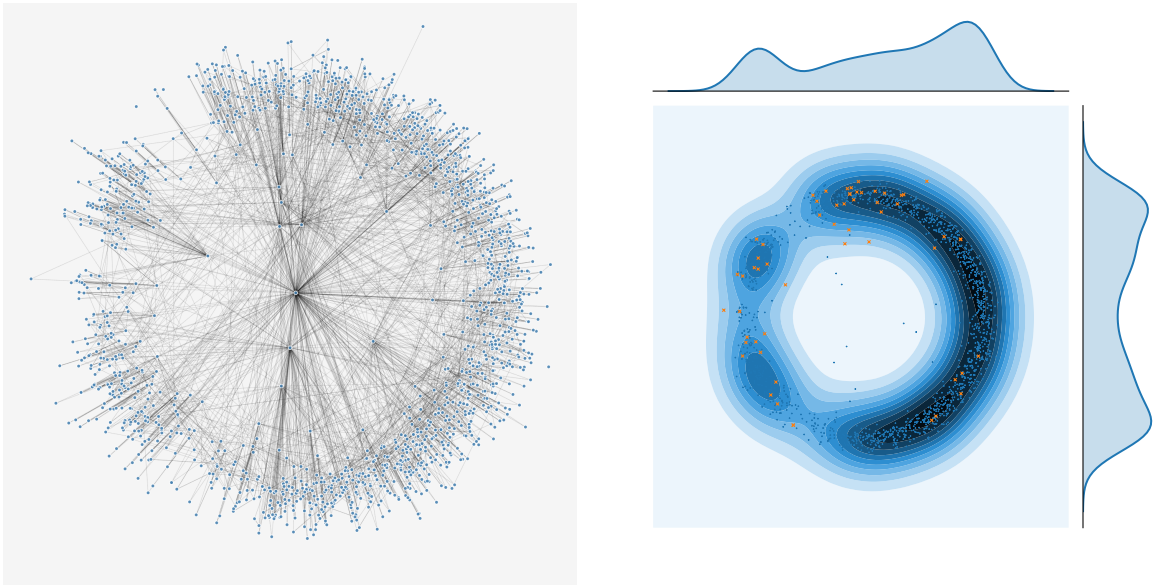
One of the oldest fields of data visualisation fall in the realm of cartography. Here the shapes and distances between points on the surface of the earth (an oblate spheroid) are mathematically mapped onto a 1D plane for graphing purposes [Thomas, 1952]. Since the process of dimensionality reduction will produce inherent distortions within the final product, we end up with a range of map projections, with each striving to achieve a different aim (Figure 1.7). The Pierce Quincuncial, for example, is a conformal mapping technique mapping the surface of a sphere to a square with minimal deviation in scale and the ability to be tessellated in all directions. The Mercator, on the other hand, is a cylindrical projection which grew in popularity due to its unique ability to represent any course of constant bearing[1] as a linear segment within the shipping and navigation industry. Finally, the waterman butterfly presents the globe as a truncated octahedron. This allows for the reconstruction of a three-dimensional world from a 2D plane (i.e. a printed sheet).

[1] Also known as a 'rhumb', or 'loxodrome', and consists of an arc crossing all meridians of longitude at the same angle.



(a) Pierce Quincucial (b) Waterman Butterfly (c) Mercator

Figure 1.7: **A selection of map projections.** These have been created using DataDrivenDocuments [?] and show a range of methods for mapping the spheroid shape of the Earth onto a 2D plane.



(a) The Mercator graph. (b) Mercator species distribution

Figure 1.8: **The Mercator Projection.** (a) represents the output from the Mercator graph layout algorithm. (b) provides a kernel density analysis of the node distribution within this. Here (a) shows graph structure by revealing the density of connections between different nodes, while (b) reveals the density of nodes at a specific location.

More recently, the mathematics of mapping a large dimension onto a simpler one has been applied to the problem of graph representation. [García-Pérez et al., 2019] uses the latent hyperbolic geometry of the Mercator layout to provide a 2D embedding for complex real-world networks. This produces a polar representation (r and θ) of the system, where relationships of related species are of the same angle (θ), with nodes of a high degree are closer to the centre (low r value, where r is the radius from the centre). Using the chemical mechanism from the APHH Beijing campaign (described above), this produces a layout, (Figure 1.8) where (a) shows the graph-based representation including links, and (b) shows the density distribution for all nodes. Figure 1.8b shows that primary emitted species (orange dots) are uniformly (radially) distributed for angles and Figure 1.8a reveals that influential nodes with a high degree (highly connected) are located close to the centre of the graph. Although the

Mercator embedding does reduce the 'hairball' problem experienced by other layouts, it does not take edge weight/direction or self-loops. This means that it works well for the representation of the general network layout, but cannot be used for advanced data exploration concerning simulation results.

1.2.2.3 Force-Directed

Force-directed graph layouts are the results of the Spring-Electrical model. This was first introduced by [Eades, 1984] and further improved by [Fruchterman and Reingold, 1991]. Force-directed layouts are, in essence, a simple physics simulation of like-charged particles representing the nodes. These particles act similarly to protons which experience Coulomb repulsion and try to get away from each other. If there is a relationship between two nodes, an attractive spring-like force is introduced, drawing the nodes back together.

In the case of a weighted graph, where each link (or relationship) has a value associated with it, we can adjust the spring coefficient of the attractive force to reflect this. This results in a layout where strongly connected objects are drawn together, and weakly connected ones further away. Uses for this type of representation have been shown biology, social networks, and with this thesis atmospheric chemistry [Muelder et al., 2014; Kohlbacher et al., 2014].

Next, we describe the Barnes-Hut algorithm, a mapping algorithm which builds a hierarchical tree of the data by splitting a plane into quartiles. This is used within the many force-directed graph layouts, including those of Force Atlas 2 and Yifan Hu, described shortly. Once this has been done a selection of four different layout algorithms shall be discussed.

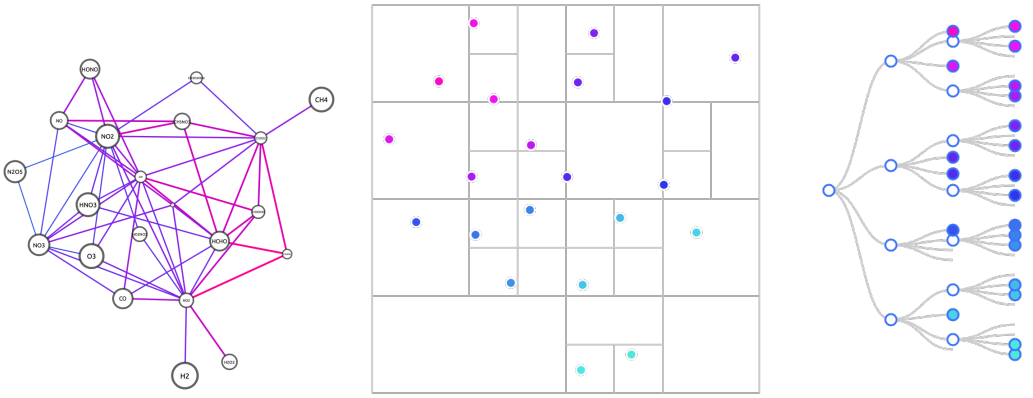
Barnes Hut Algorithm

Since calculating the attractive/repulsive forces for each node of a large graph can be computationally intensive, many force-directed layouts rely on the Barnes-Hut approximation. This solves the N-body problem of pairwise reactions between nodes, $O(n^2)$, by approximating long-range reactions by grouping such nodes and applying a single action on their centre of mass- reducing the computational time to $n \log n$.

To do this, first, a spatial index of each node is constructed (see below). This can either be done using a quadtree (2D) or octree (3D). Following this we calculate the centre(s) of mass, allowing us to approximate the repulsive forces of a force-directed graph.

Quadtree Construction: A quadtree is the recursive partitioning of two-dimensional space into a set of quadrants (a set of 4 squares). This process is repeated, with each square then being divided into four itself, until there is only a single point within a cell. This converts a network, into a hierarchical

tree representation of the nested quadrants in which each point resides (a quadtree), Figure 1.9.



(a) Methane Graph (b) Quadtree constructed from Figure 1.9a

Figure 1.9: **Demonstration of the formation for a quadtree from a force directed graph of Methane (including inorganics)**. (a) shows the force directed graph of Methane from which the quadtree has been constructed- edge colours represent the flux between species. Here we partition the area into 4 and start at the top-leftmost cell. This is then partitioned into 4 itself in a recursive process until there is only one point per cell. We repeat the process to any remaining cells in a clockwise manner (b). The hierarchical tree (b right) shows the containing structure for each node. Here the colours represent the order in which nodes have selected (starting at pink and ending in blue).

Having defined this, we move on to looking at the graph layouts.

Force Atlas 2

The force atlas two [Jacomy et al., 2014] algorithms is a force-directed layout designed primarily for scale-free[2] network spatialization. It is primarily designed for the use of networks consisting of 10 to 10,000 nodes and uses barns-hut approximation for the calculation of forces. Attractive forces are derived from the spring-electric model ($F_a = -k.d$), where k is the spring constant and d is the distance between the two nodes. Optional features for the graph include dissuasion by degree (separating nodes with a high number of total links/reactions), logarithmic attraction forces, adjustable gravity (attraction the centre of mass of the system to prevent disconnected components from drifting away) and collision detection to prevent overlapping nodes. Finally, an adaptive cooling scheme is applied, where the overall energy of a system is gradually decreased, allowing the nodes to settle into a low energy state.

Yifan Hu

The Yifan Hu graph layout [Hu, 2004] is a multi-level graph drawing algorithm which uses the Barnes-hut algorithm with an octree layout. As with the force atlas algorithm, Yifan Hu also has an adaptive

[2]A network whose degree distribution follows a power law (7 degrees of separation). This is described in Chapter 1.

cooling aspect to it - meaning that as the algorithm is run its energy is progressively reduced, allowing the system to settle within a low energy state.

The main difference within the algorithm, however, is the use of the multilevel approach. This has been applied to graph partitioning [11,12,23], matric ordering [24] and the travelling salesman problem [5]. This works by graph coarsening (coalescing neighbouring nodes and weighting them), running the algorithm on the course graph, prolongation and then refining the results. This produces an algorithm that runs faster than the Force Atlas, however, is constrained to only working on un-directed edges.

OpenOrd

A force-directed graph algorithm capable of scaling to very large graphs [Martin et al., 2011]. OpenOrd uses simulated annealing (see below), which has five distinct phases. These are each run for a fraction of the total number of iterations and mimic the different states experienced when heating/cooling a physical object (liquid, expansion, cool-down, crunch and simmer) - hear each state describes the amount of energy assigned to the nodes within the force simulation. In addition to this, the OpenOrd algorithm applies a degree of edge-cutting to remove a percentage of edges experiencing the most stress within the physical system. This allows the network to open out into a more aesthetically pleasing layout.

Simulated Annealing

Most iterative layouts are updated interactively from some initial configuration in an attempt to reach the lowest energy state of the system. In most cases this results in a minimum configuration; however, this is generally a local minimum rather than the desired global minimum (the optimum low energy state of the entire system) [Davidson and Harel, 1996]. To overcome this, the work of Metropolis et al. [1953], which was later formulated in general terms by [Kirkpatrick et al., 1983], was used to lay the foundation for simulated annealing algorithms.

Annealing is usually used to describe the slow cooling applied to liquids for them to reach a crystalline (totally ordered, minimum energy) form. It can be shown that if the atoms(nodes) are cooled too rapidly (losing energy quickly and coming to a quick stop), they will form amorphous structures representing the local minima, as opposed to the desired global one. If cooled slowly, our graph is allowed to find a thermal equilibrium at every temperature. A slow cooling constant is applied, whilst occasionally supplying the system with short bursts of energy, that may allow it to overcome local minima.

tsNET

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction technique which mimics the style of a force-directed graph (this is discussed in Subsection 2.3.3). tsNET[3] is a graph drawing algorithm which leverages the non-linear dimensionality reduction capabilities of the t-SNE algorithm [Maaten and Hinton, 2008a]. This works by first computing the shortest-path distances between all nodes to produce a distance matrix. This distance matrix is then used to construct a cost matrix which consists of the sum of three terms:

1. A measure of the divergence between picking pairs of low- and high-dimensional data points.
2. A compression factor, known to reduce the t-SNE optimisation time, taken from [Maaten and Hinton, 2008b].
3. A repulsion term to prevent nodes clumping together.

Node positions are then determined by the minimisation of the cost matrix using gradient descent - an optimisation algorithm used to minimise a function by iteratively moving in the direction of the steepest descent.

Note: Although tsNET makes for an excellent alternative to classical graph layouts, it does not take link direction into account.

1.3 Selecting The Best Graph Drawing Layout.

Subsubsection 1.2.1.1 explained the importance of removing overlapping edges and Subsubsection 1.2.1.2, the desire of having a well-distributed graph layout. This subsubsection builds on those criteria, assessing all the graph layouts described within this section (Mercator, Force Atlas 2, Yifan Hu, OpenOrd and tsNET). These all use the chemical mechanism representing species within the APHH campaign in Beijing [**?**]. Here we look at the distribution (Subsection 1.3.1) and density (Subsubsection 1.3.1.3) as they affect a users ability to isolate the shortest path (fastest flux).

Force-directed graphs place a greater emphasis on node positions,

Criteria, such as the ability to isolate the shortest path (in this case the fastest flux), are essential in determining the usefulness of a graph. Comparing different layouts [Pohl et al., 2009] found 68% of user-chosen routes to reflect the shortest path between them.

[3] A play on t-SNE and network.

This is due to the force-directed layout placing a greater emphasis on node positions and distance than other layouts. For comparison, the same study found this to be 40% for hierarchical layouts and only 2% for orthogonal ones. This section compares some graph layouts and their effect on user readability.

1.3.1 Graph-Node Distribution

Subsubsection 1.2.1.2 explained the importance of node distribution within a graph visualisation. Additionally, Purchase et al. [2003] explains that if we partition the viewing medium into quartiles, and populate each quadrant with an equal number of nodes (homogeneity), this drastically improved the usability and symmetry of a graph.

The problem is that for complex real-world graphs is that they often contain nodes with many reactions between them (Figure 1.10). Such regions of dense, indecipherable links (often referred to as hairballs [Ma and Muelder, 2013]), obscure nodes and edges within a region, making it impossible to read.

Methods such as edge pruning [Dianati, 2016] (removing unimportant links) can be used to reduce complexity. This process may be done either post computation (syntactic representation) - resulting in a loss of information, or during the algorithmic approximation (e.g. within the OpenOrd algorithm) - where any removed edges are then re-introduced in after the final layout has been generated.

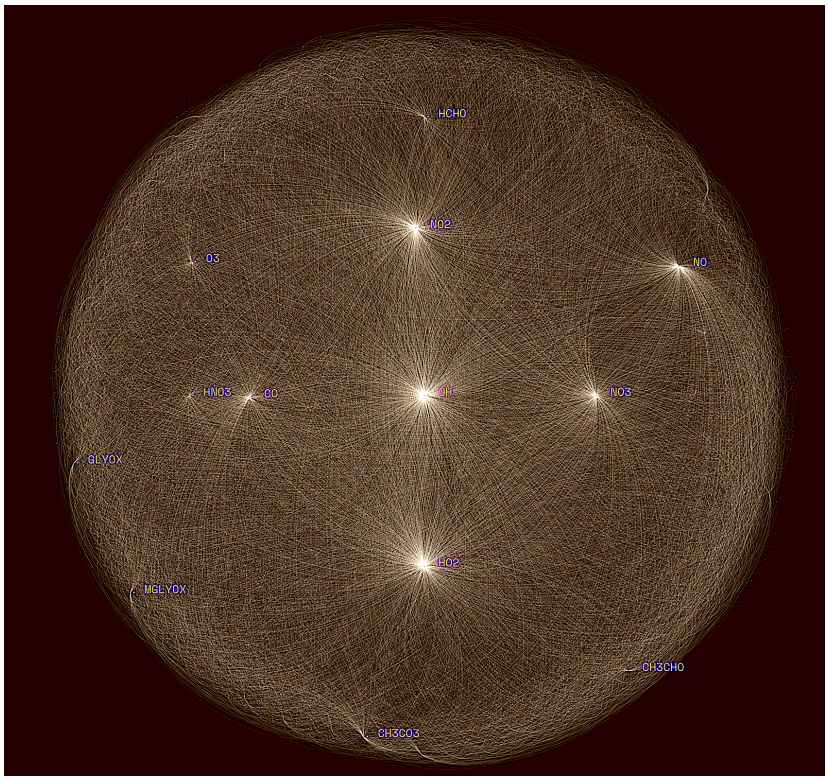
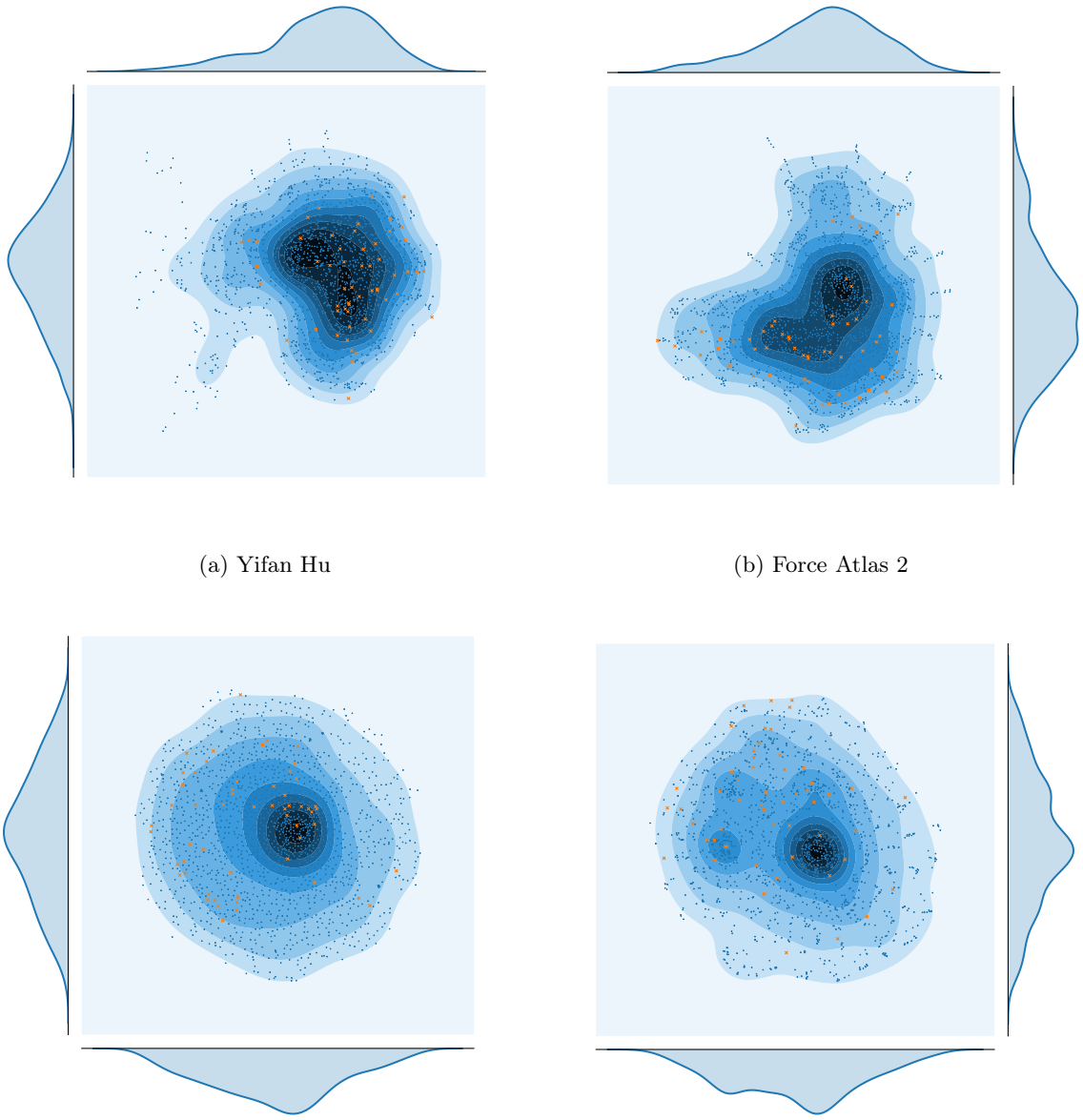


Figure 1.10: **A graph of the full MCM - a hairball.** The high number of nodes and edges (especially those to inorganic species), causes a high degree of obfuscation, rendering the graph unusable. Species with a large number of reactions (links) are labelled.

1.3.1.1 Evaluating Node Distribution For The Beijing Mechanism

In deciding which layout algorithm produces the best graph-node homogeneity, a kernel density approach is used to compare node distributions across 2D space in Figure 1.11, and the Mercator density plot from earlier (Figure 1.8b).



(a) Yifan Hu

(b) Force Atlas 2

(c) OpenOrd

(d) tsNET

Figure 1.11: **Contour and kernel density plots showing the node distribution for different graph layouts.** Line charts show the distribution of nodes in the x and y directions, while the contours represent density with respect to the location of each node (the crosses). Primary emitted species are coloured orange, and the darker contour polygons show areas of higher density.

This style of plotting allows for the easy location of areas containing a large number of nodes (high density) through the use of the contour-colour gradient. In an ideal graph, we would have groups (clusters) of high density, all of which would be evenly dispersed around the 2D plane. Additionally,

the use of x/y kernel density can show us the homogeneity of a graph - a perfectly homogeneous (lattice) graph will have a uniform distribution across both axes. For a modular graph of evenly dispersed groups, we would expect an oscillatory distribution of similar amplitudes. Using this criteria, the Mercator (Figure 1.8b), tsNet (Figure 1.11d) and Force Atlas (Figure 1.11b) score the highest, where the OpenOrd and Yifan Hu graphs containing a gaussianesque distribution across both axes - a distribution conducive to the production of a hairball.

1.3.1.2 Distribution Of Primary Emitted Vocs

Within the construction of an atmospheric chemical mechanism, a chemist first begins with a primary emitted species. This is then broken down to produce other species, depending on its structure and functional groups (**??**). This process suggests that in constructing a network from such a mechanism, this structure will be prominent. Knowledge dictates that a chemical graph should start from a large emitted species, and aim towards carbon monoxide (and ultimately CO_2 although this is not included in the MCM). To show such a structure, we expect any primary emitted species to be evenly distributed and the chemistry to tend towards the location of CO (the centre). In searching for a layout that satisfies this requirement, the tsNET graph (**??**) is found to be the best, followed by the OpenOrd and ForceAtlas2. Yifan Hu (Figure 1.11a) and Mercator (Figure 1.8b) both contain areas where many of the primary emitted (orange) species are grouped and are therefore unsuitable for the representation of the MCM structure.

1.3.1.3 Calculation Of Spatial Clustering

Subsubsection 1.3.1.1 explains that the ideal (modular) graph consists of many groupings of like chemistry evenly scattered across the graph. This requires a degree of regular anisotropy to produce 'clusters' of densely connected nodes, sparsely separated in space. To calculate this, we can rely on Voronoi tesselation.

Voronoi Tesselation

Voronoi Tesselation is the process of finding the largest area closest to a specific point. It can be thought of as a container with a bubble at the location of each node, where each bubble collapses to fill the largest area possible.

Here we begin by partitioning the graph plane into the same number of cells as our nodes. Next, each cell polygon boundary is calculated such that all the points on it lie closer to its seed (origin) node than any other. Mathematically these are referred to as the perpendicular bisectors of the lines

between all points.

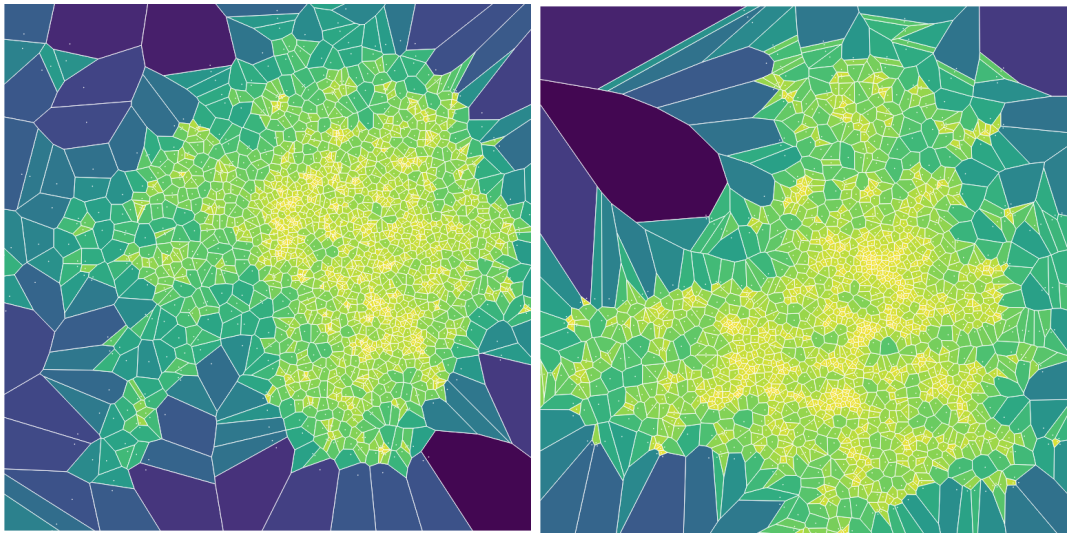
Finally, we calculate the areas of each polygon and use it to represent the density distribution between neighbouring nodes of our graph. To simplify the visual analysis of each graph, these are also coloured in Figure 1.12.

Visual study of node clustering

The method of using Vornouli tessellation for the calculation of density has been used in the study of neurones [Duyckaerts and Godefroy, 2000] and areas of fixation when viewing images [**?**]. We apply it to the nodes of a set of force-directed graphs to determine the graph layout, which provided the best high-low density ratio for the atmospheric chemical mechanism of a Beijing environment.

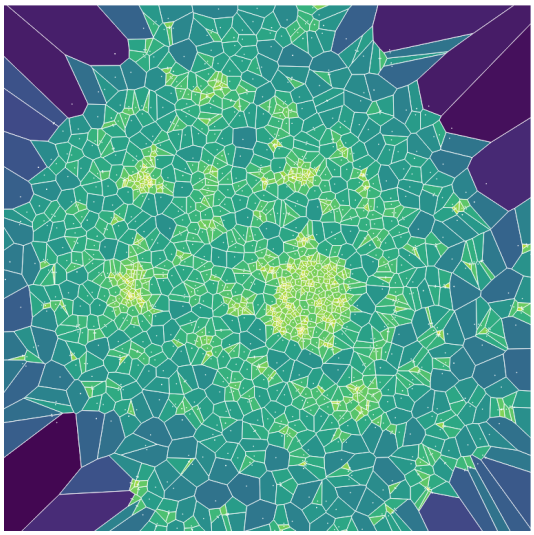
Using Vornouli tessellation in Figure 1.12 we find that the OpenOrd and tsNET (Figure 1.12d,Figure 1.12c) layouts have the highest isotropy - containing cells of similar sizes and consequently a small colour gradient. This suggests that these are spatially efficient layouts as they do not reveal any additional information about nodes of similar chemistry.

In contrast, the Mercator layout, despite having a high $x-y$ node distribution, contains large areas of unoccupied space due to its non-linear density distribution. Using this measure of spatial modularity, we find that the ForceAtlas2 and YifanHu (Figure 1.12b,Figure 1.12a) graphs have distinct modules of high density distributed across the entire graph, which is what we expected from the MCM network.

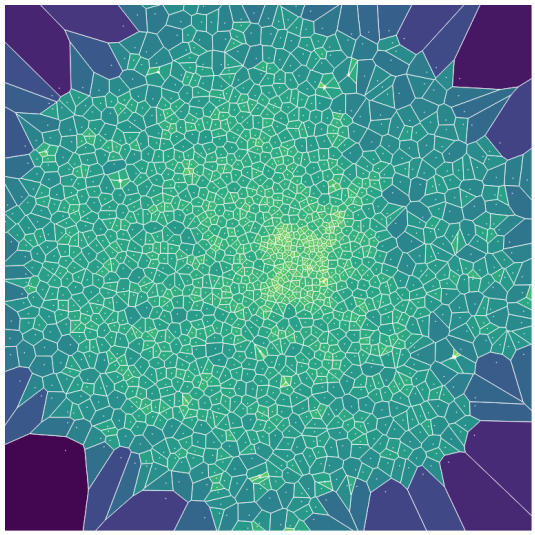


(a) Yifan Hu

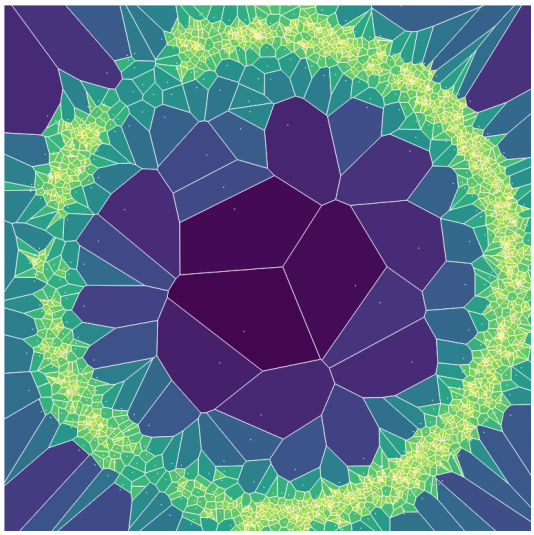
(b) Force Atlas 2



(c) tsNet



(d) OpenOrd



(e) Mercator

Figure 1.12: **A visual analysis of node-cluster density using Voronoi tesselation.** Each polygon is centred on a node - its area represents the space between the node and its nearest neighbours. Colours follow the normalised size of the Voronoi cells/polygons.

Mathematical Analysis And Layout Selection

In contrast to the qualitative analysis of the visualisation, it is also possible to calculate the distribution of polygon areas (and this node dispersion) of each graph through the use of a boxplot (Figure 1.13), in addition to the minimum, maximum and outlier properties, a boxplot allows for the comparison for the interquartile range (IQR) between graphs. If these values are large, they signify a more significant distribution between sparsely and densely grouped nodes (a commodity that is desired). The medians location within the IQR can also be used to indicate the dominant size of the polygon within the graph. Here a greater number of smaller polygons is desired and can be seen by a median approaching the lower box boundary (25^{th} quartile).

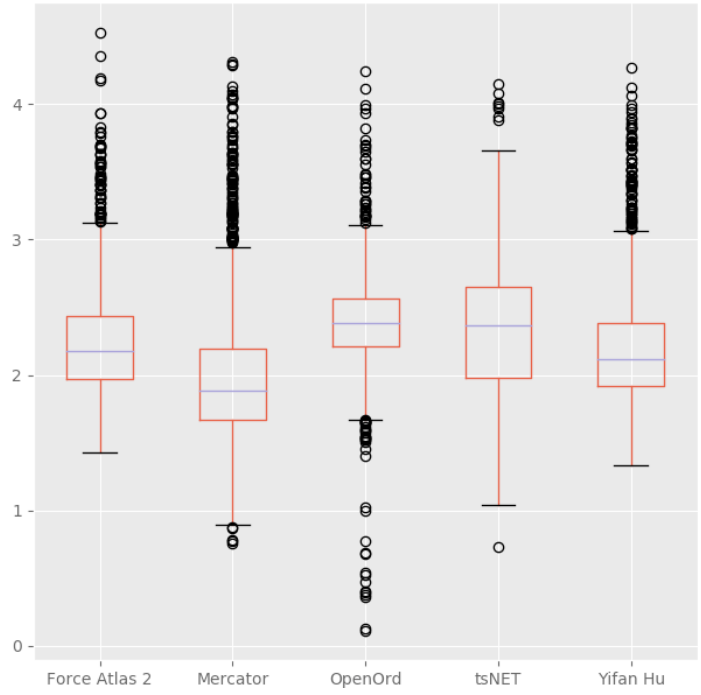


Figure 1.13: **Voronoi $\log_{10}$(Area) BoxPlot for all plots in Figure 1.12.** This provides a mathematical analysis for the areas around each node within a graph.

Figure 1.13 shows Mercator to provide the best result. However, it is noted that although the boxplot contains the best ratio, its radial shape is not conducive to the representation of modularity within a chemical network. tsNET contains the largest IQR, and since it is not able to handle the directed edges of an atmospheric chemistry graph, this again has to be ignored. Finally, although OpenOrd can reduce the number of hairballs within a graph by using simulated annealing and edge-cutting, its homogeneous isotropic node distribution (small IQR with a sizeable median value) make it not most effective at highlighting the structure of the MCM. Yfan Hu layout fares better with regards to the box plot, yielding an overall lower box, with a similar IQR and median ratio. Here its lower median suggests more high-density nodes, with a similar distribution to the ForceAtlas2. This makes

sense since the two algorithms share many similarities; however, the inability to handle directed edges makes it unsuitable for our application.

By process of elimination, this leaves the ForceAtlas2 algorithm as the best candidate for representing a chemical mechanism. Its directed nature, coupled with intuitive design make it applicable and easy to explain while maintaining the ability to produce a clear representation of any underlying structure. In addition to this, its uniform spatial distribution (Subsection 1.3.1) makes it a better candidate than the Yifan Hu graph (which fared slightly better in the boxplot).

1.4 Graph Semantics

Deciding the correct semantic (relating to meaning) representation for visualisation is often just as important as selecting the correct syntactic (structure) style. Semantic features are often applied post generation [Bennett et al., 2007] and are used to encode additional information and clarify the data. As a means of achieving both an aesthetically pleasing outcome, and an easy to understand visualisation, we must first consider what features we, or the reader, are most interested. Once this has been decided, we begin to explore various methods for representing them.

1.4.1 Limitations

Before selecting any semantic features, we must inform ourselves of the visual, cognitive and technological limitations of the visualisation, medium or user.

Visual

In visual analytics, the most significant bottleneck falls on the resolving power of the eye - this is known as an acuity (sharpness or clarity of vision at a distance). Acuities are a measure of the angle of an observed object with the viewer's eye using arcs (one arc minute equates to $\frac{1}{60}^{th}$ of a degree). This provides a unit of measurement for the total amount of information density we can feasibly perceive [Ware, 2013].

In ophthalmology there exist four types of acuities:

- **detection**: The smallest size an object can be whist still being shown
- **recognition**: The smallest size an object can be to be recognised
- **resolution**: The smallest distance between two objects before they begin to merge

- **localization**: The smallest amount of visual change that can be measured between two objects

These provide a set of considerations which may be used to assess a visualisation. Depending on what encoding we use, it is possible to improve/hinder the reader's ability to perceive information, (Figure 1.14). An example of this would be that for a Macbook Pro retina screen[4], where at 87 pixels/cm we can resolve at most 2 million resolvable nodes (at 57 cm from the screen). If we wished to add links between nodes, the total items identified is reduced to one million [Jankun-Kelly et al., 2014].

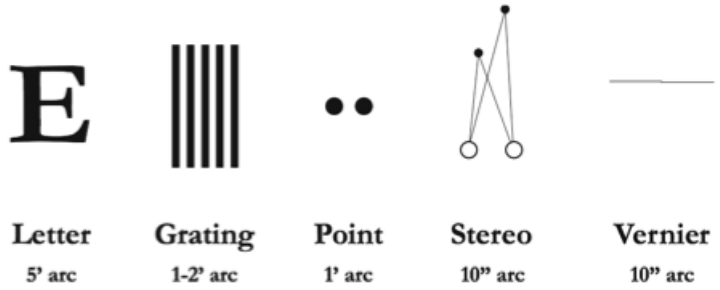


Figure 1.14: **Important acuities in visualisation.** Here a double prime " represents an arc minute which equates to an angle of 1/60 of a degree. A single prime ' is that of an arc second with is 1/60th of an arc minute (or 1/120 of a degree). For comparison the maximum angular resulution of the human eye is stated as 28 arc seconds [Deering, 1998] - this means we can only ever see up to 28 nodes or 2 verneers (disjoint lines) at any one time. Source: [Jankun-Kelly et al., 2014; Ware, 2013]

Cognitive

Although it may be possible to distinguish 1 million nodes and links visually, interpreting and understanding these presents another problem. The visual thinking laboratories [VTL, 2019], have a range of publications exploring how presentation can improve though, cognition and communication between info-graphic and reader. Steven Franconeri [2018], explains that the time required to interpret a visualisation is directly related to the encoding used to highlight the data within it. Also 'intentional blindness'[5] and misinterpretation are problems which are often occurred with poorly thought out encodings.

In considering the cognitive load of a visualisation Norman [2005] provides a list of three categories which should be explored:

1. Firstly, we have the visceral level, a subconscious process where decisions are made rapidly based on sensory inputs to the body. This is usually due to our inherent ability to locate patterns and

[4]A retina screen, is half the maximum possible resolution of the human eye at a 30cm distance. Additionally, the operating system interpolates in sets of 4 pixels, such that the image displayed may not be at full resolution.

[5]The failure of a user or audience to notice a fully visible feature because their attention was engaged by something else - e.g. misdirection in magic tricks.

changes due to semantic properties which shift the focus of the user.

2. Next follows the behavioural level (mostly subconscious). These are often learned reaction to changes noted as part of the visceral level. Here reactions may be honed on and influenced by past experiences and events.

3. Finally, we reach the reflective level. Here the user collates all sensory input from the previous two levels and makes an informed conclusion about the underlying data. Conclusions drawn here can be used to bias the methods used within the behavioural level in future events.

Technological

In addition to human limitations, there may be restrictions due to the medium a visualisation is created/presented. In addition, to monitor resolution, much scientific research is constrained by the size, resolution and colour quality of the presentation mediums used for talks, printing or posters. Ware [2013] explains that a printer capable of producing 1200 dots per inch squared, can only do this for black/white binary images. If for instance, 256-greyscale is used, the resulting resolution is then at-least ten times smaller. This is because printers a Monet (dot matrix) style approach to create shading and colour. It follows that at full colour[6], the output resolution will be worse.

It is also essential to have a graph fitting the same overall shape of the canvas on which it is presented [Taylor and Rodgers, 2005]. This not only makes optimal use of any space available but also reduces the visual complexity as it minimises the number of distinct shapes available to the user.

1.4.2 Node Encoding

Within a graph, the nodes or vertexes are a representation for the set of items we have an interest. In addition to the relationships between them, items often contain a multitude of features which describe them. Examples of these may be useful information for a person, categories of characters or the chemical composition/concentration for a species in the MCM.

Each of these additional properties can contain valuable information for the interpretation of the graph, and the interactions between nodes. It is, for this reason, that graph convoluted neural networks [Klicpera et al., 2018] require a 'feature matrix' describing each node, in addition to the network structure and edge weightings.

This subsection addresses a number of ways in which additional information can be encoded in the representation of a node, Figure 1.15. Each different category colour matches the title description in

[6]CYMK (Cyan Magenta Yellow Black)

the text.

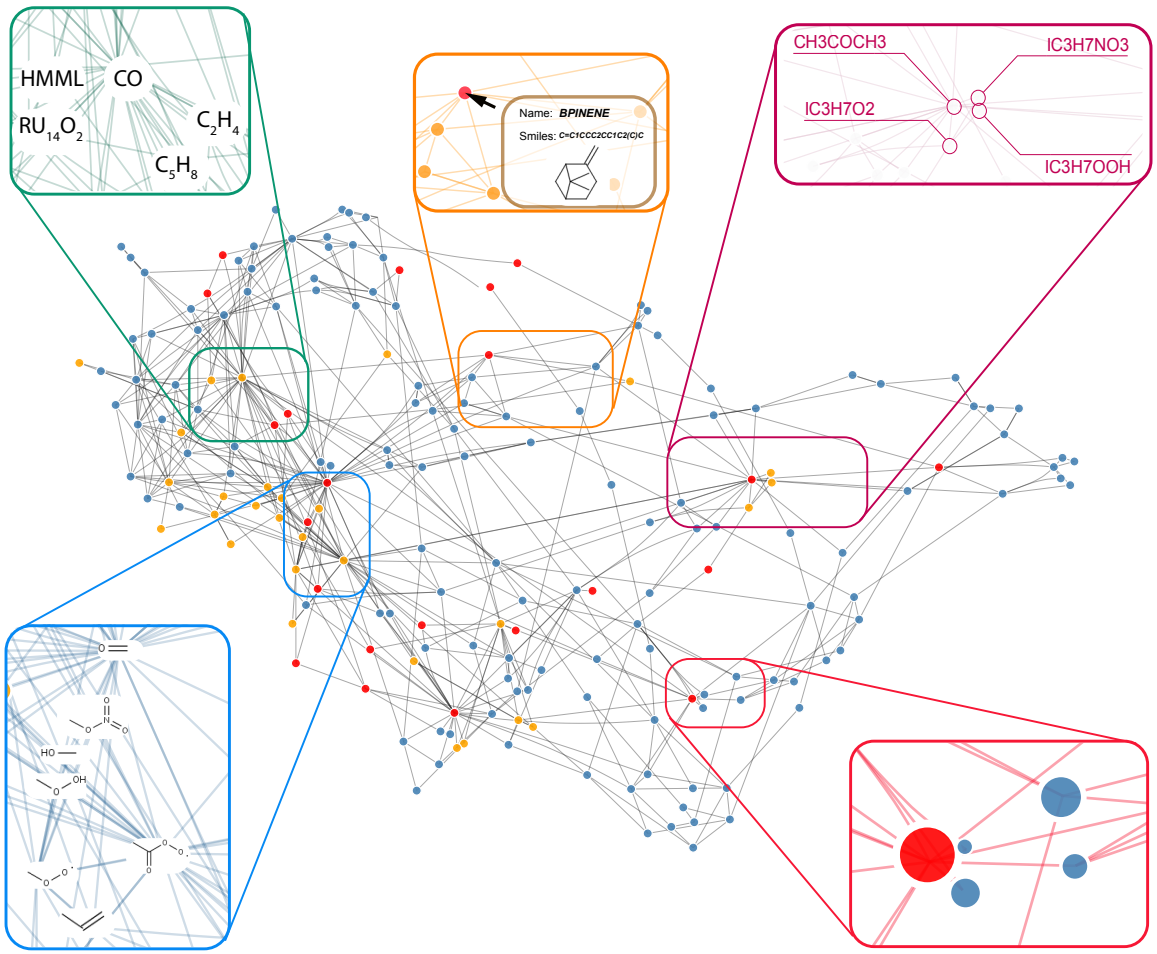


Figure 1.15: **A graph showing 5 different node encoding methods.** These are Circle Atrributes (red), Chemical structure (blue), Species Name (green), External Labels (maroon) and interactive selection (orange). The network shows the Common Representative Intermediate species [Jenkin et al., 2008] mechanism. Node colours represent primary emitted VOCs (red), MCM species (orange) and lumped CRI-only species (blue).

Circle Attributes

The simplest of these range from the use of colour and stroke (outline) to shape/size as a way of indicating a group. Here it is possible to provide information such as a species concentration based on its size, its importance with its colour, its degree with its opacity and its category with its stroke colour [**??**]. Such decisions depend on what properties the user is trying to show. For instance, red species in Figure 1.15 are primary emitted VOCs, orange species exist between both the MCM and the CRI (see figure caption) mechanism, and blue ones are lumped species which do not appear as part of the MCM.

Chemical Structure

Traditional chemical diagrams use the chemical structure to depict the node (Figure 1.2). This makes it intuitive to extract information about the functional group and potential bond changes within species. Such a method of representation, is indeed useful, however when visualising hundreds, or thousands of nodes on a page, it results in occlusion, or labels being too small to resolve visually.

Species Name

Much like the chemical structure, a species name is proven useful in explaining to the user its chemical structure or properties (often due to prior knowledge, or the ability to look this up). Unfortunately, since names have differing lengths, this can cause problems, especially with large numbers of closely located nodes. A solution to this may be to adjust the font size to fit in within the circle radius of the node. However this does come with its problems - for instance, tiny nodes may have text smaller than a pixel, or the misleading notion that longer names are less important since they are represented by a smaller font.

Interactivity

Ben Shneiderman, one of the first, and most prominent, researchers in human-computer interaction coined the phrase *'overview first, zoom and filter, details on demand'* [Shneiderman, 1996]. This is the philosophy behind most data orientated interaction design and can be applied to graphs. One example is the selection only of reactions relative to a node of interest, as is shown in Figure 1.16.

For complicated systems, interactivity plays a vital role in unravelling complexity and reducing clutter [Shneiderman, 1997]. This is a method which lessens the cognitive load (Figure 1.4.1) on the user, allowing them to query only items of interest, whilst still displaying all the information in a single location [Görg et al., 2007]. This fits in Ben Shneiderman [1985]s 8th rule of interface design, which explains that people only ever remember *'seven plus or minus two chunks'* of information.

A comprehensive list of all available interaction types and styles are provided by Wybrow et al. [2014]. Some examples of interaction are:

Hi-lighting

- Hovering
- Brushing and Linking
- Magic Lenses (see hidden objects)

Navigation

- Pan / Zoom
- View Distortion (fisheye)

Visual Structure-Level Interaction

- Selection
- Changing layout/mapping attributes
- Changing representation

Data Level Interactions

- Adding / Filtering
- Search / Query

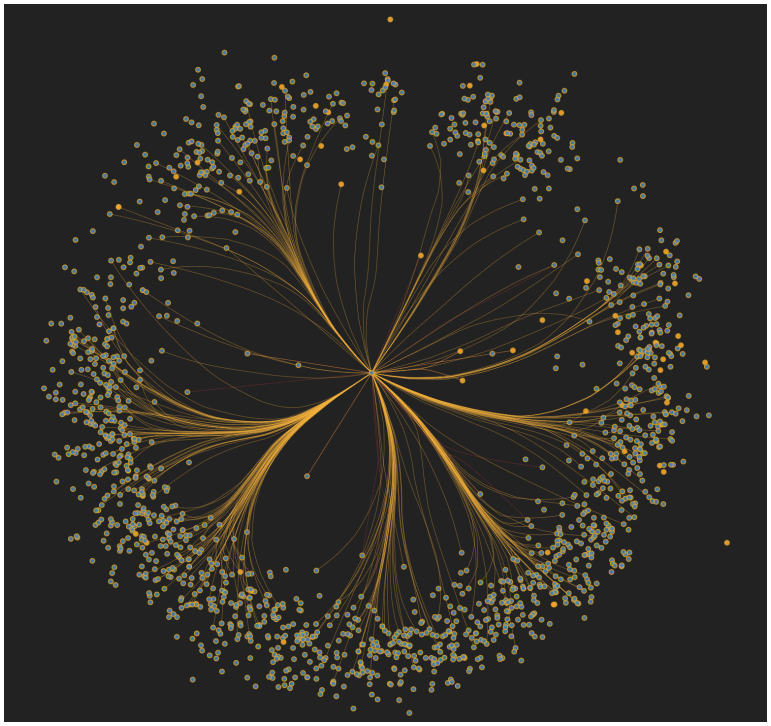


Figure 1.16: **Using mouseover edge-selection to highlight all links related to a node.** This figure shows how in using interactivity it is possible to reduce clutter and filter the information presented by a densely populated graph. In this case, the Mercator projection (Subsubsection 1.2.2.2) is used, with reactions relating to Carbon Monoxide (centre) highlighted. Orange lines represent reactions producing CO whist the red (some of which may be hidden) are of reactions with CO.

External Labeling

In cases where interactivity is not possible, such as papers, books and this thesis, an alternative approach to data selection has to be employed. Here nodes which are central to the explanation of a certain point are filtered by the author and displayed through the use of external labels. It is found that having links at 45 and 90-degree angles (such as in transport maps) lead to a clearer layout and better distinction from the links already within the graph. Automatically generated labels within the thesis are made using Lu [2019].

1.4.3 Edge Properties

Defining the purpose of force-directed (graph-energy) models as a means for creating a visualisation from which the viewer can infer properties of the data [Noack, 2004], it can be shown that this criterion is easily met in small and sparse graphs. However, non-planar examples with high edge density (lots of links) can easily result in tangled results with impractical running times [Kumar and Garland, 2006]. In most cases attaining an optimal solution here seems to be computationally infeasible [Davidson and Harel, 1996]. This is generally because graphs primarily focus on hi-lighting a specific purpose or following a set of aesthetic heuristics [Pohl et al., 2009].

1.4.3.1 Muti-Variate Edges

Since there are multiple relationships between species, it is important to decide if simplifying the network would be of benefit. Although it is possible, multiple edges may cause unnecessary clutter for larger networks. Instead, it is often useful to simplify the number of edges in the network and encode the edge properties within the vector object. This allows the user to retrieve any additional information by hovering over the edge or connecting nodes, as required. Should the topic of interest require a specific property, then it would also be possible to remove, or hide, all edges which do not contain it. This produces an interactive graphic containing all the required information, as and when needed, without the unnecessary clutter of having every reaction shown.

1.4.3.2 Edge Direction

When using a directional graph, it is the convention to use arrowheads to represent the direction of flow. However, in high-density regions, it is often found that arrowheads take up precious real estate in the drawing area [Dwyer et al., 2006a]. As an alternative, colour and line-type can be used to represent the direction instead - this was seen in Figure 1.9a (the quadtree example). This example can be shown in the routing networks presented by [Di Battista et al., 2004]. One example applicable for chemistry would be the use of dashed lines to represent mono-directional relationships and continuous lines for bi-directional ones.

1.4.3.3 Edge Shape

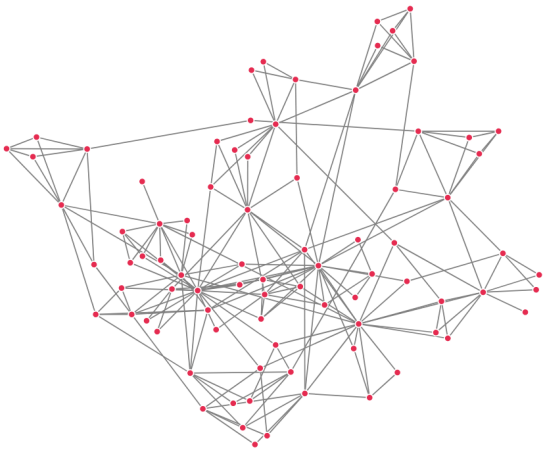
Edge shape is essential, as it is the medium we use to represent relationships within a graph. For orthogonal graphs (Subsubsection 1.2.1.1), multiple lines are used to simplify the complexity of a graph and reduce edge crossings [Di Battista et al., 1994]. In increasing the number of lines within an edge, this multi (poly)-line graphs can be modified to give drawing with nicely curved edges. In a

similar manner, it is possible to replace the edges in a straight-line graph with Lombardi-style curves or cubic beziers [Chernobelskiy et al., 2012; Goodrich and Wagner, 1998].

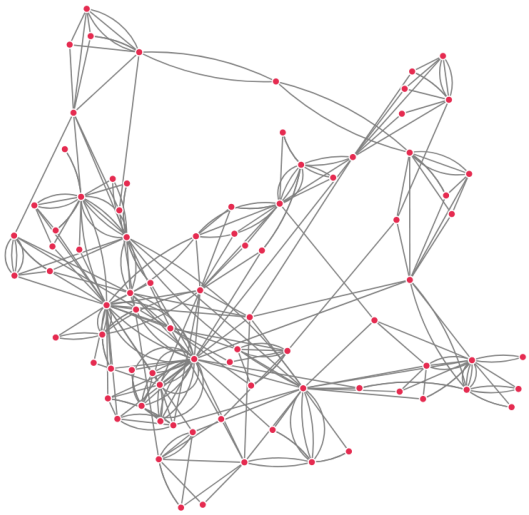
Using a mechanism describing the reactions of Butane in the MCM, we compare many edge shapes (Figure 1.17). In this graph, each node has multiple edges between nodes (multi-link). Each edge represents a different reaction between the species. Figure 1.17a shows the straight-line representation of each graph. Since each edge represents the shortest distance between two nodes, any additional reactions pairs between similar nodes are hidden from view. If using this type of representation, it is advised to take the net edge direction and weight between the values. An improvement to the straight-line graph comes from the use of quadratic curves (Figure 1.17b). This allows for the symmetric distribution of edges within the graph. If instead an asymmetric representation for each edge is required, it is possible to use a bezier curve (described below) instead of a quadratic edge (Figure 1.17c). These can provide additional information through the use of control points, which can alter the shape, steepness and asymmetry of each link.

Bezier Curves

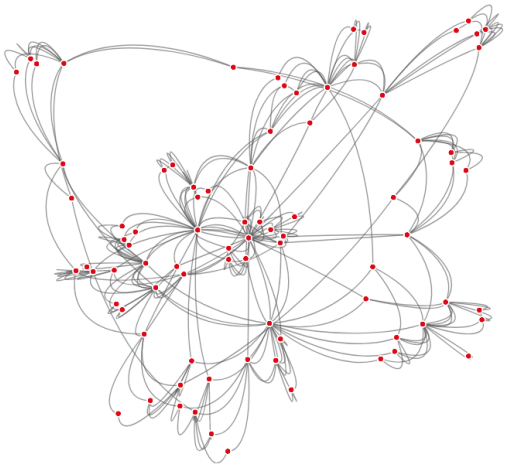
Bezier curves are named after Pierre Bezier who used them in the bodywork design of Renault cars in the 1960s [Hazewinkel, 1997]. Since then they have been widely used in graphs, computer graphics, font design and animation/interactivity response [Goodrich and Wagner, 1998; Hazewinkel, 1997; Mortenson, 1999]. Bezier curves come in a range of possible dimensions; cubic beziers are the most commonly used within network visualisation. These contain four control points respectively, which can be used to determine the shallowness of the curve through design. In general, relatively shallow curves are prefered, as these do not introduce unnecessary edge crossing or abrupt changes, which have been shown to hinder a users ability to isolate items of interest [Purchase et al., 2003].



(a) Linear (single-edge)



(b) Quadratic (multi-edge)

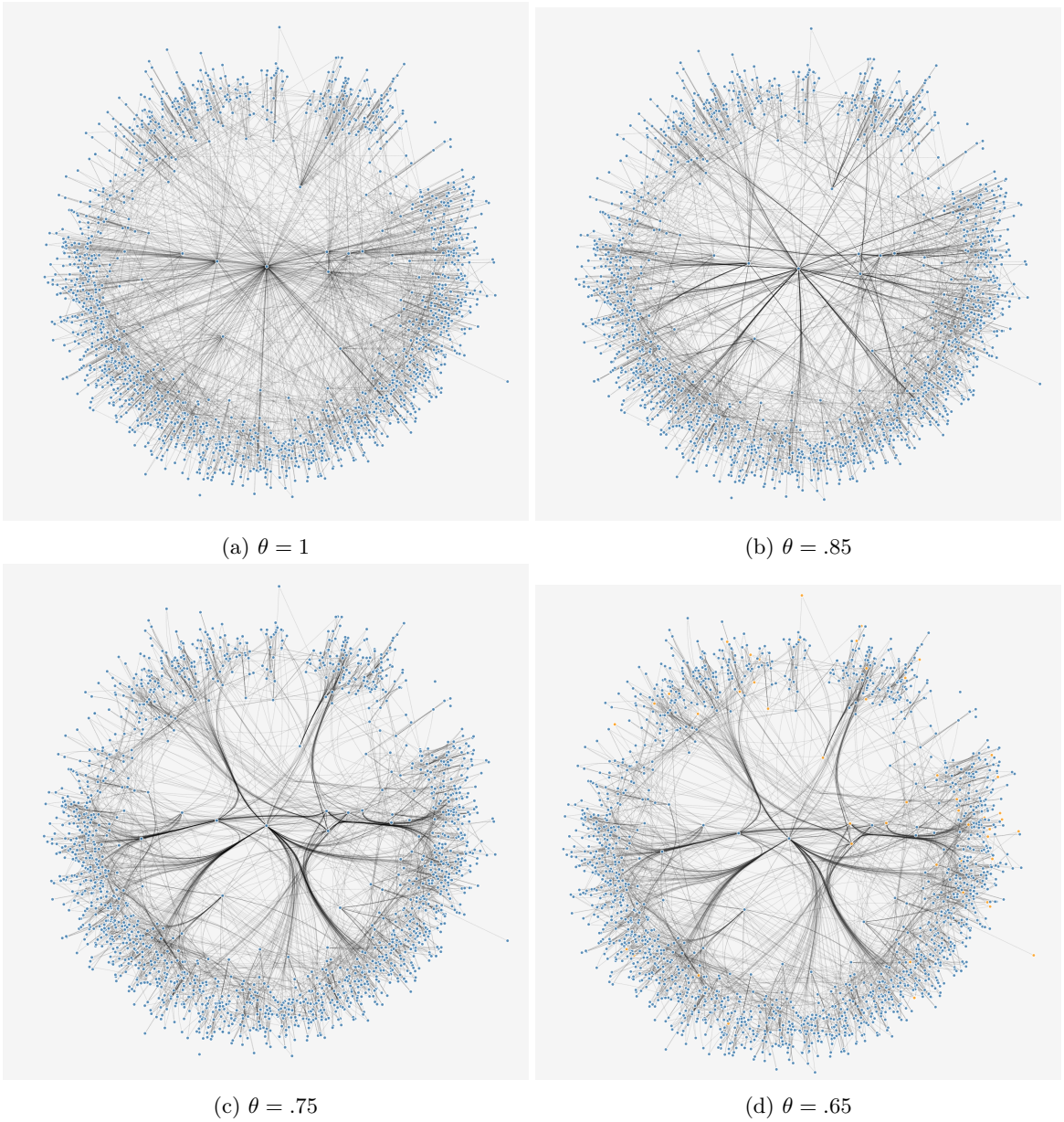


(c) Bezier (multi-edge)

Figure 1.17: **A selection of edge shapes for the butane network.** These show linear (a), quadratic (b) and bezier (c) edge shapes for the same network. In general the bezier curves appear to provide the shapes of the most aesthetically pleasing graphs.

1.4.3.4 Edge Bundling

Pioneered by Holten [2006], edge bundling techniques are an effective way to reduce visual clutter. Much like a force graph, edges are represented as a string of lined points. This allows for edges to be pulled together (attracted to one another) and produces a visualisation akin to moving water droplets on a hydrophobic surface. Figure 1.18 shows how in changing the amount of attraction between edges, it is possible to reduce clutter in a visualisation.



(a) $\theta = 1$ (b) $\theta = .85$

(c) $\theta = .75$ (d) $\theta = .65$

Figure 1.18: **How the compatibility threshold affects edge bundling using the Mercator graph from Subsubsection 1.2.2.2.** In increasing the amount, edges are attracted (θ) it is possible to improve the clarity of a graph. However, there reaches a point where this distortion can worsen the result, confusing the reader, or creating a false positive. For this reason, I generally use only a slight bundling value > 0.7.

1.4.3.5 Power, Routing And Confluence.

Confluent graphs use a graph drawing method in which edges are not drawn as individual distinguishable geometric objects, but rather as a crossing free system of arcs and junctions. [Förster et al., 2019]. Their design is similar to that of the edge bundling algorithm, except that rather than bundling edges spatially (a design which may introduce ambiguity), the bundling is done based on connectivity and can help reduce clutter by grouping multiple edges where the all target nodes are also connected to all the source nodes (Figure 1.19) [Bach, 2020].

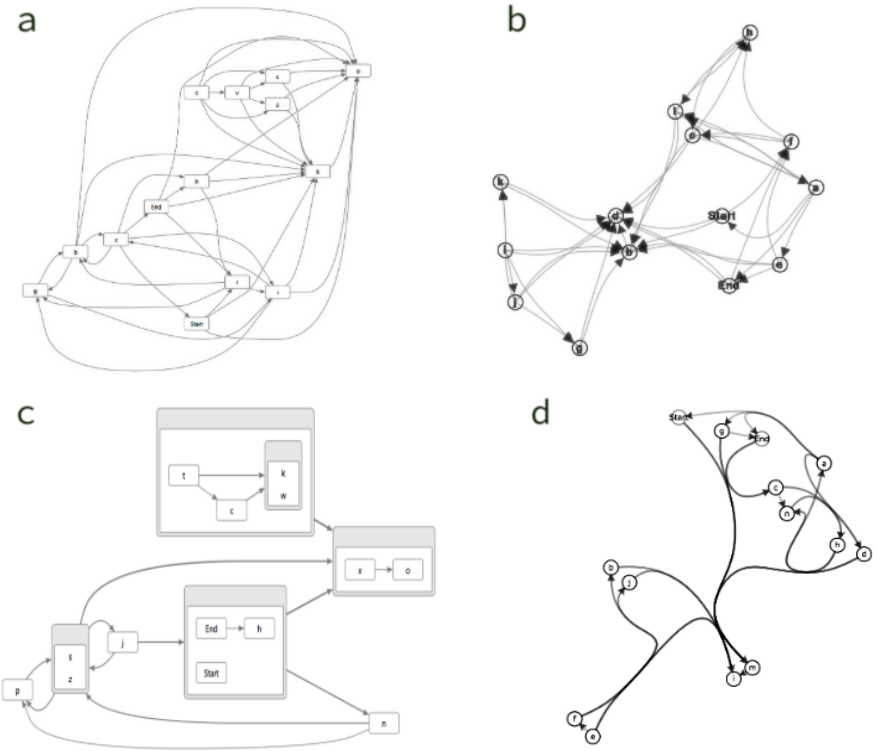


Figure 1.19: **An example of confluent bundling.** A traditional network (a), Edge bundling (b), Power Graph (c) and Confluent graph (d) representations. Source: [Bach, 2020]

Using the oxidation of butane as an example, we shall explore the construction process of a confluent. Starting with the network presented in Figure 1.17, we create a power graph - power graphs are a representation of complex networks where sets of items identical source and target links are lumped or grouped within a single item. Next multiple edges which follow the same path are bundled through a 'routing' node to create the routing graph in Figure 1.20. The newly created routing nodes are now used as control points for the mapping of the graph using basis-splines[7] (Figure 1.21). Finally, any crossing links are removed, leaving the confluent graph in Figure 1.22.

Confluent drawings have been found to have many applications (e.g. the ego-centric author network

[7]These are similar to bezier curves but require a degree (p), $n+1$ control points, and a knot vector of $m+1$ points. Knots are the things that make the curve continuous

and social interaction graph), they generally perform best in sparse networks with locally dense clusters of a tree-like structure [Bach et al., 2017]. Although sparse, the cyclic nature of atmospheric chemistry does not allow for a sufficient reduction in complexity to make them a suitable improvement over traditional graphs. The use of very close-fitting basis-splines in addition to a routing graph (confluent graph with crossing artefacts), may, however, help to simplify specific layouts or mechanism subsets with a certain amount of tweaking.

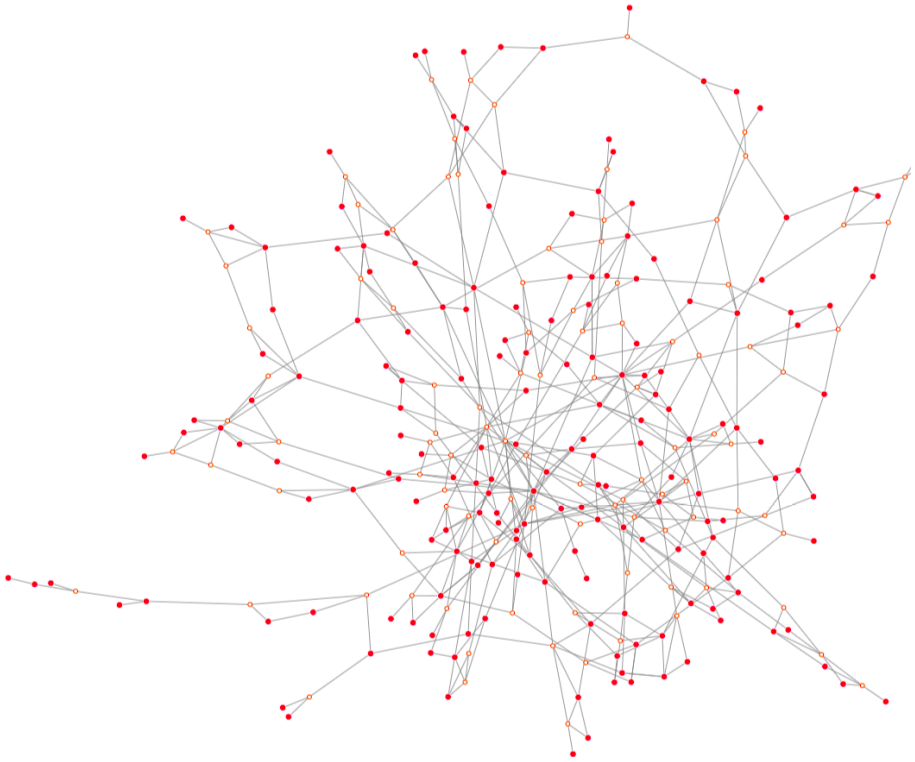


Figure 1.20: **The routing graph of the butane mechanism.** Here paths which contain two or more bundles have an extra 'routing' node introduced (orange stroke)

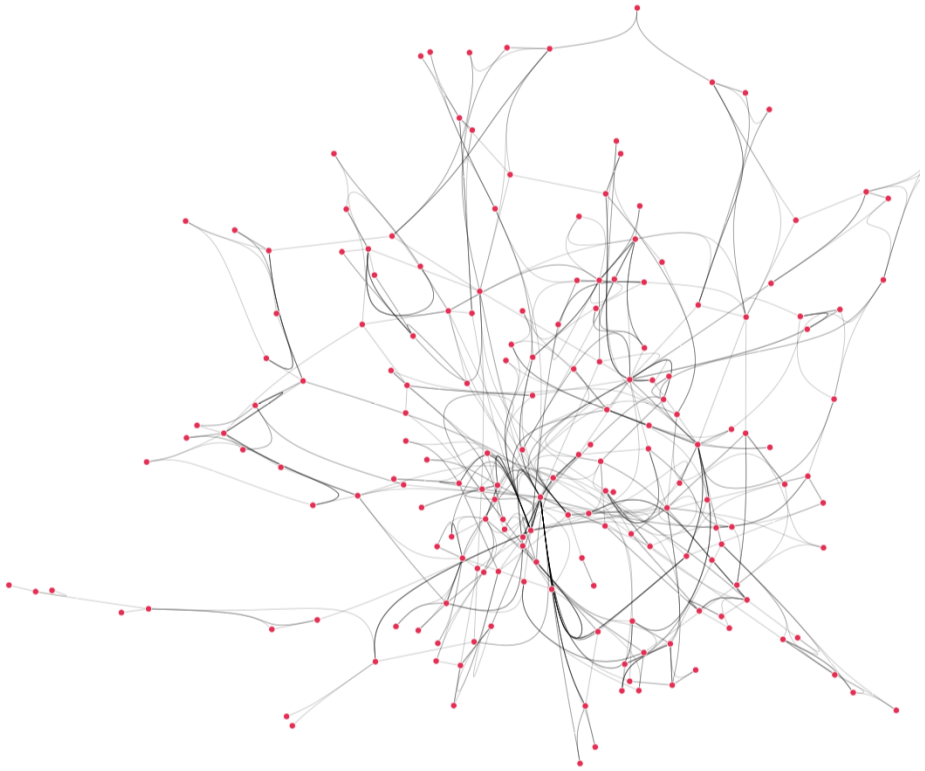


Figure 1.21: **Confluent graph with crossing artifacts.** The routing graph with the addition of basis-splines using the orange routing nodes in Figure 1.20 as control points.

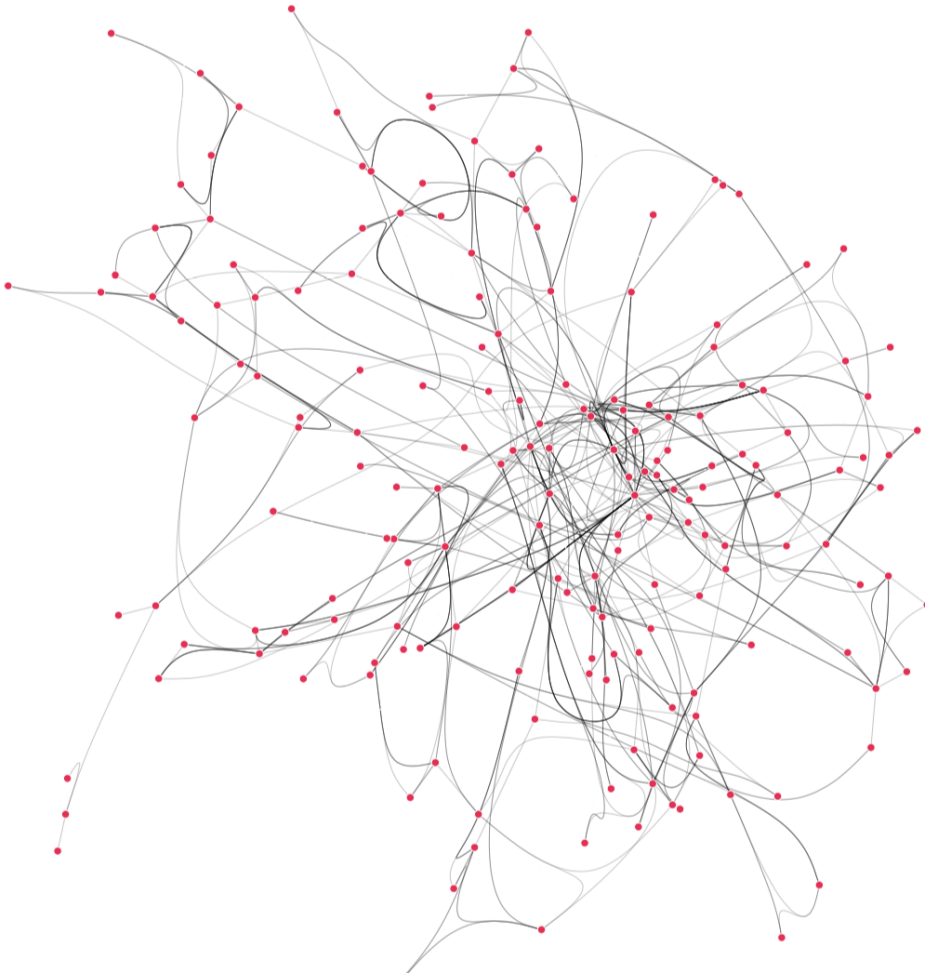


Figure 1.22: **Confluent graphs without crossing artifacts.** The remaining confluent graph with crossing edges removed.

1.4.3.6 Angle / Continuity

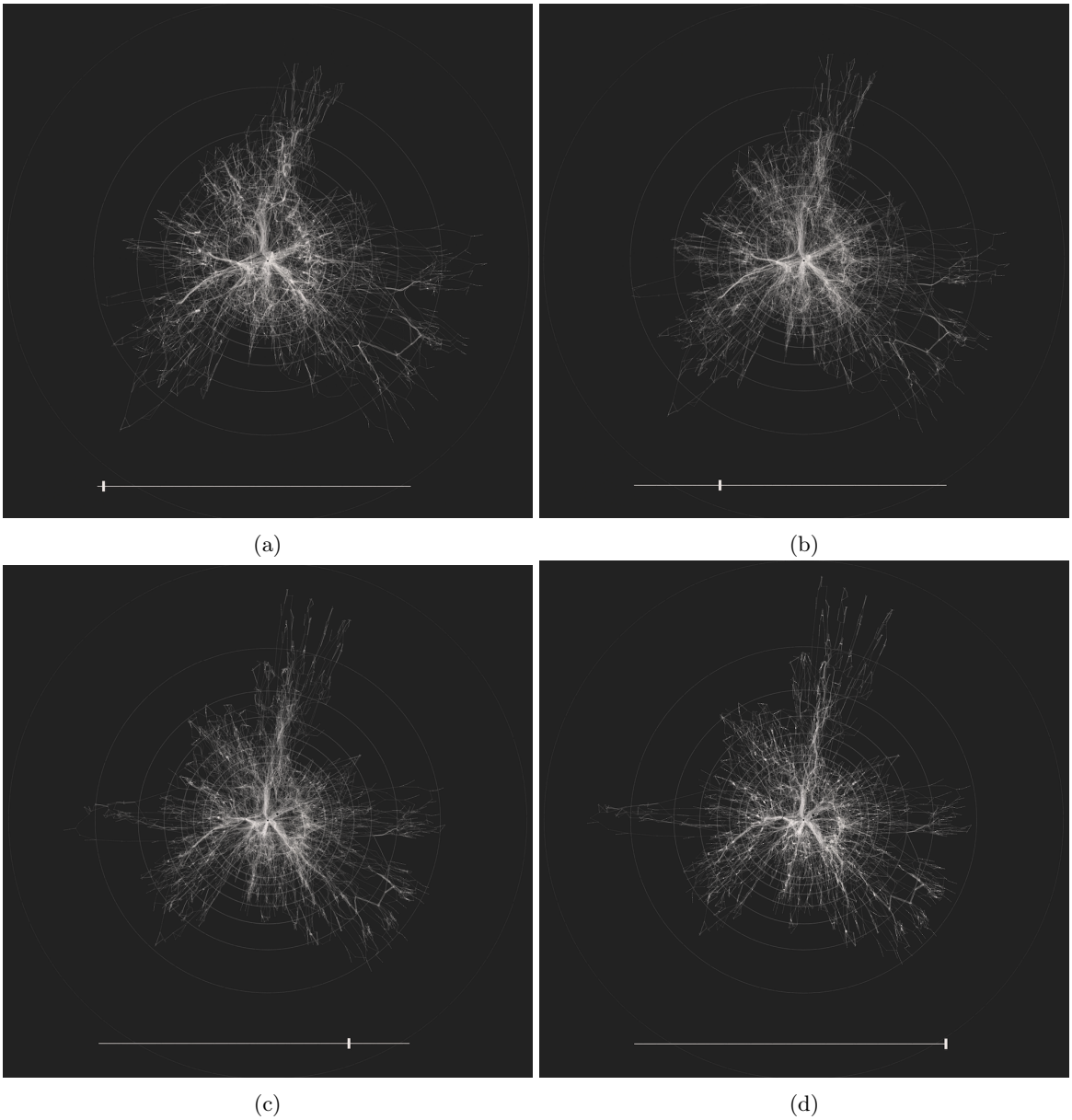
Visual representation utilises our conscious and unconscious pattern recognition and intuition abilities [Dixon, 2012]. To avoid apophenia (finding patterns where they do not exist), careful consideration has to be placed in the design of a graph layout. Although edge crossing is often thought of as the most import aesthetic metric, finding continuity between incoming and outbound edges of a node was found to be if equal importance [Ware et al., 2002].

Reducing the angle between related edges increases readability and allows the behavioural process to infer information about a graph correctly. This process can be compared to predicting the direction of turbulent vs laminar flow. In addition to this edges should be spaced evenly around the node, maximising the minimum-edge-angle between all edges of a node [Bennett et al., 2007].

1.4.4 Temporal Projection

Story-telling has been an effective method to convey information, experience and cultural values for almost as long as people have been around. Many real-life physical processes occur over time and thus allow the use of a story-telling analogy. Gershon and Page [2001] provides a generic structure which begins with creating a general overview of the subject. Events are then animated in order of occurrence and defined as we go along. Finally, any remaining conflicts and uncertainty are addressed, and these are rectified. Using this as a template for our graphs, we find that the content is usually given in the form of a title or figure description, the evolution as the visualisation, and finally the reflection and resolution through the use of user interaction (e.g., node hi-lighting, zoom or animation).

Since very few graph layouts support dynamic time-varying graphs [Kumar and Garland, 2006], several methods of visualising temporal events have been developed. Although storylines can be useful for drawing the evolution of simple systems, these break down when dealing with large numbers of dependant variables. Force-directed layouts may be adapted, to suit these better, whereupon the initial positions of the previous node endpoints are used as the initial positions for consequential simulations. Three methods of representing these are shown in [Ellis, 2018] - screenshots of which are shown in Figure 1.23.



(a) (b)

(c) (d)

Figure 1.23: **Film style representation of temporal changes in a network.** Showing the temporal changes from a model simulation of the Beijing atmosphere. (a) shows a weighted graph at midnight. With the addition of daylight, the chemistry speeds up, causing the force graph to contract, changing the overall network shape (the faster reactions have a stronger attractive force). The animation of this can be found at `https://github.com/wolfiex/DanEllisThesis/blob/master/daynight_26mb.gif`

Finally, user-interaction such as hi-lighting key nodes/links, zoom and animation[8] may be used to clarify information at the reflection stage.

[8][Archambault et al., 2014] notes that animation poses high demands on the user's visual memory and that snapshots are likely to miss underlying patterns. For this reason, interactive techniques that can allow retrospective selection of timesteps allows for a good compromise between these.

1.4.5 Additional Dimensions

Additional dimensions can be used to emphasise certain aspects of our graphs. For instance, multiple layers may be used in a directional graph to separate the importance of the nodes [Dwyer et al., 2006b]. Figure 1.24 shows the first, second and third-generation species of a mechanism containing isoprene in three dimensions, where each layer in the third direction represents a different generation of species. Such visualisation may be explored interactively, with the aid of a computational input device (a mouse, keyboard or device gyroscope), or with the aid of red-cyan 3D glasses (for non-interactive mediums such as print).

Different layers can be used to separate primary VOCs, from species which result in their production (+1 layers) and loss (-1 layers). Temporal data (e.g. Figure 1.23) can also be presented in this format. The only drawback is the high possibility of obfuscation which may result from many layers of overlapping information.

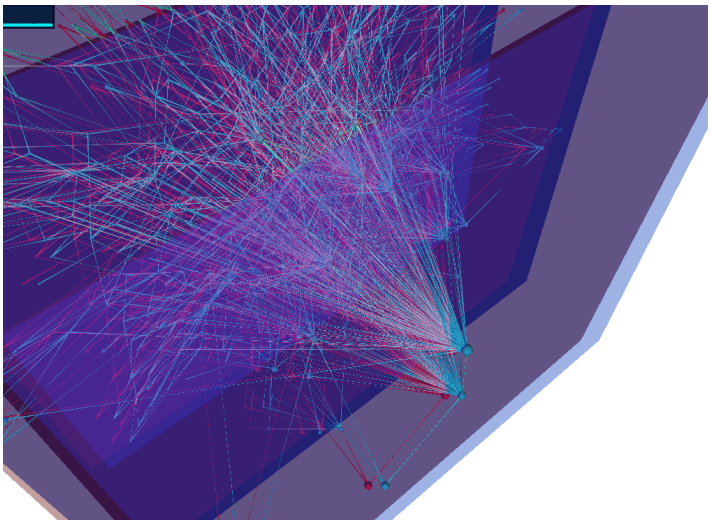


Figure 1.24: **A 3D representation of a graph to hilight certain features.** The first, second and third generation species of isoprene shown as an interactive 3D anaglyph.

1.4.6 Summary

In this section, the different semantic methods have been explored. It is found that additional information such as concentration or functional groups may be represented as node size/colour and that there are a range of edge plotting styles that may be used to reduce clutter.

In the next section, we combine both semantic and syntactic representation and apply it to an atmospheric chemical mechanism. Using the graph visualisation tools described above, we access the quality of information which may be inferred through representing a mechanism in this way.

1.5 A Chemistry Case Study

To conclude, we apply many of the tools described above to a single case study. We select an MCM subset representing the VOCs measured as part of the APHH campaign in Beijing [**?**]. The chemistry of the mechanism is initiated using the conditions in **??**, and propagated by the Dynamically simple model of atmospheric chemical complexity (DSMACC) [Emmerson and Evans, 2009; **?**]. We run this forwards to steady-state and extract the flux between species on noon. The edge weight is the net flux (product of the species concentration multiplied by the rate of reaction for all reactions between those species and products), normalised to a value between 1 and zero.

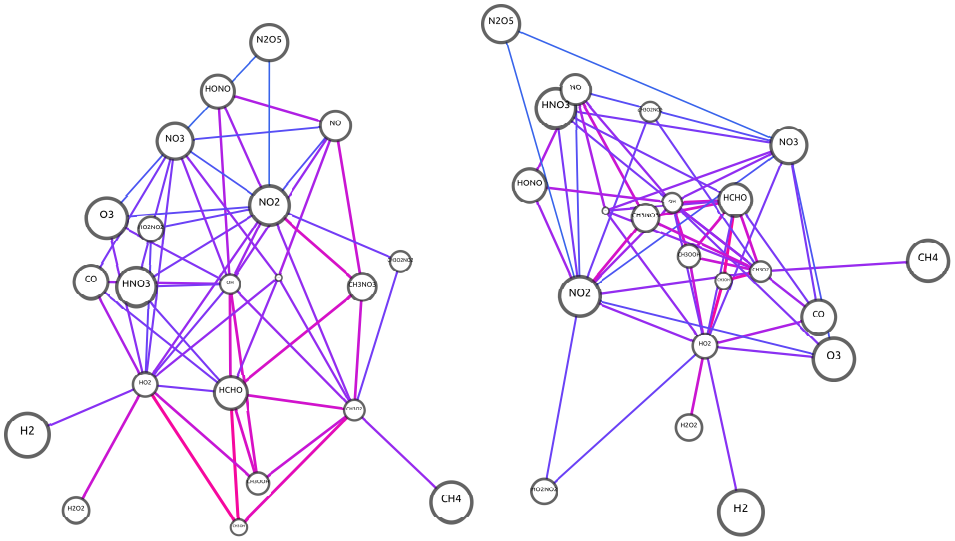
1.5.1 Semantic And Syntatic Considerations.

1.5.1.1 Syntatic Representation

Since we shall be using simulation data, we require a layout which deals with both direction and edge weights. The spring-like description of the ForceAtlas2 is chosen from Section 1.2. This feature highlights fast reactions by bringing nodes together. Such a property has been observed to help users select the shortest path within a network [Pohl et al., 2009]. Here users picked the shortest path an average of 68% for force-directed graphs, compared to 40% for hierarchical and 2% for orthogonal layouts. Such properties can help us locate any trends in fast reactions which may control the chemistry within a system.

1.5.1.2 Example Semantic Representation Using A Methane Mechanism.

Since the graph generated by the methane mechanism contains only a handful of species, our screen real-estate allows the listing of names for each node. Node sizes are scaled to represent the concentration of each species at that time point, and edges are coloured to represent the strength of each relationship between them. Here pink edges represent a fast-flux and blue ones a slow one. In comparing the change in graph shape between a weighted Figure 1.25b and unweighted Figure 1.25a graph, we can see that nodes connected by a high edge weight (fast-flux) are drawn closer to each other than those with a slow flux.



(a) Connected unweighted (b) Connected weighted (flux)

Figure 1.25: **A weighted and unweighted force diagram of the methane mechanism.** Here it is seen that upon weighting, edges with a faster flux (pink) are drawn closer than those of a weaker one (blue).

1.5.2 A Model Of Beijing

To perform a sensitivity study on the initial positions of nodes within the force atlas algorithm a graph consisting of links and weightings is constructed using a box model simulation of the Beijing summer environment (mid-day) and feed it the gephi software [Bastian et al., 2009] - an open-source software designed for the exploration of networks. We then script the java code to perform the functions in Figure 1.26. As part of this, nodes are initiated with a random position; the ForceAtlas2 layout is then run and then the graph is rotated and translated such that it is centred around carbon monoxide and has a 45-degree angle between this and formaldehyde. This step constrains the general orientation of the graph, allowing us to analyse the generated graphs for global and local minima. The final step is to save a copy of the generated graph layout and repeat to generate a data set, a subset of which is shown in Figure 1.27. These are discussed further in Subsubsection 1.5.2.1.

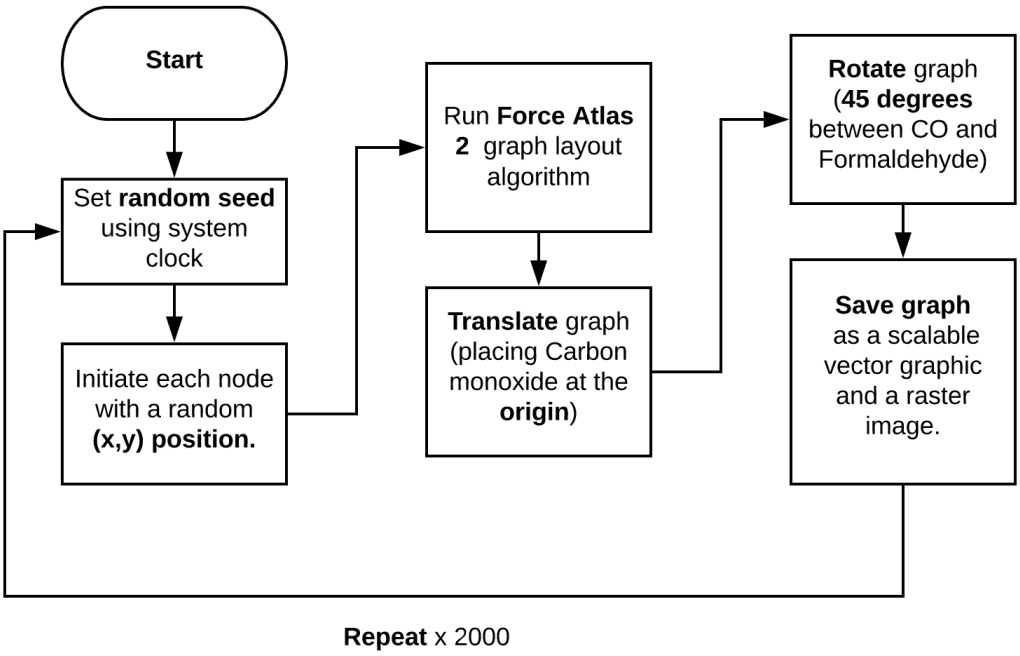


Figure 1.26: A flow chart of the process performed by the custom gephi script used to generate the data set

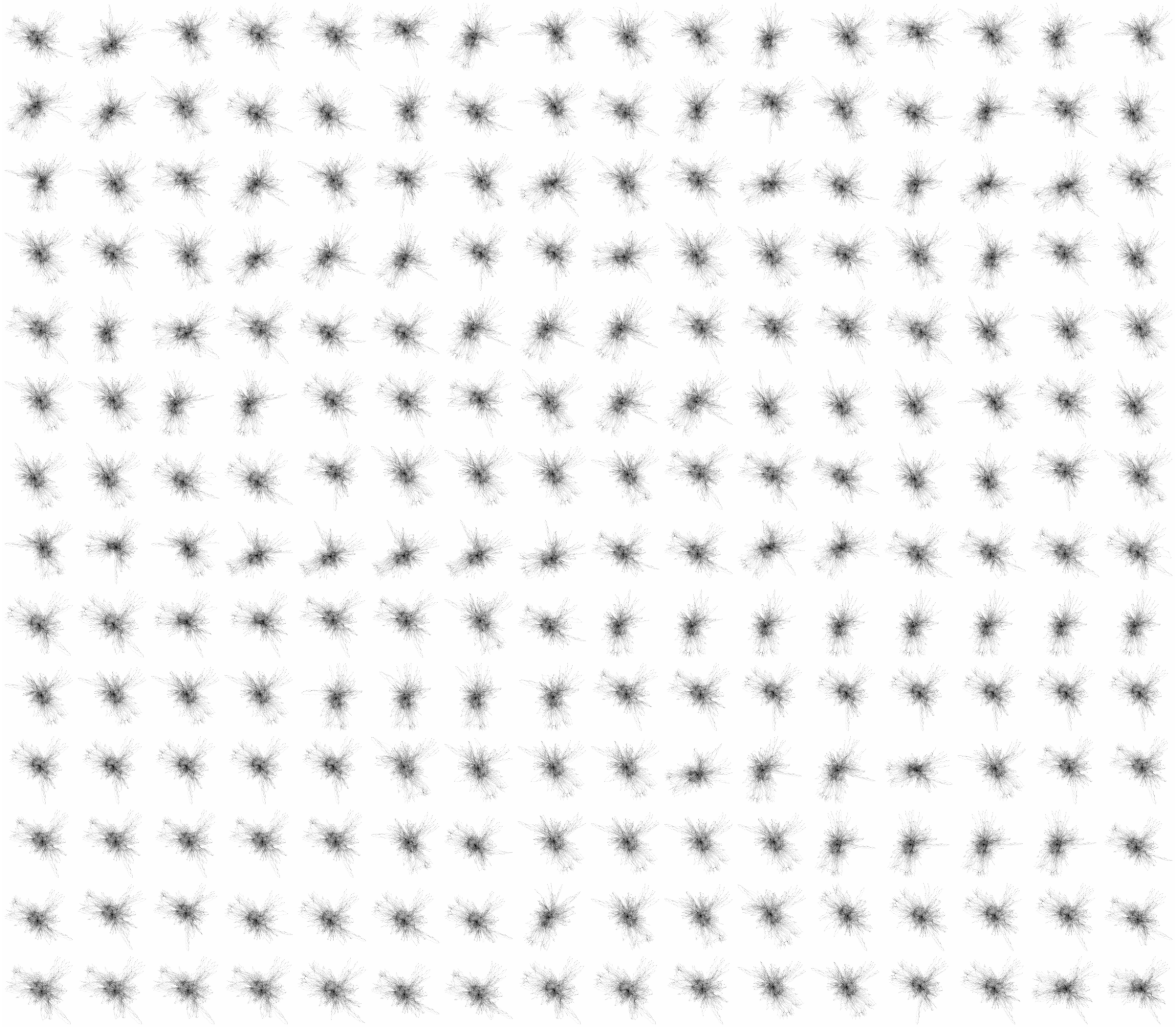


Figure 1.27: **A sample of 224 (out of the 2000) graphs generated using the ForceAtlas2 algorithm.** These represent the conditions of a spun up simulation of Beijing at noon. The shapes of each graph, and general shapes are discussed in Subsubsection 1.5.2.1 and Subsubsection 1.5.2.2.

1.5.2.1 Similarity Between Graph Shape

Although through the use of manual intervention, it is possible to perform a superficial level of shape analysis, our cognitive capabilities do not allow us to perform this task for all the simulations of Figure 1.27- less so the entire 2000 graphs in the dataset. To overcome this problem, we rely on a method of machine learning called t-Distributed Stochastic Neighbor Embedding (t-SNE) - described in Subsection 2.3.3 and is the foundation of the tsNET layout algorithm. This is a dimensionality reduction technique used in the automatic categorisation of images or photographs [Stefaner, 2020; Sangkloy et al., 2016].

The input for the t-SNE for each dataset is a flattened (1 dimensional) representation of the pixels in the image - we start and by taking a binary matrix representing each image, split it up into rows, and glue these together. The pixelmap for each image is then fed into the t-SNE algorithm from the

Scikit Learn package [Pedregosa et al., 2011]. This reduces the logical list of pixels for each image into a two-dimensional representation of their similarity. We plot each file, for its (x, y) coordinate, and isolate clusters of similarity using density contours in **??**.

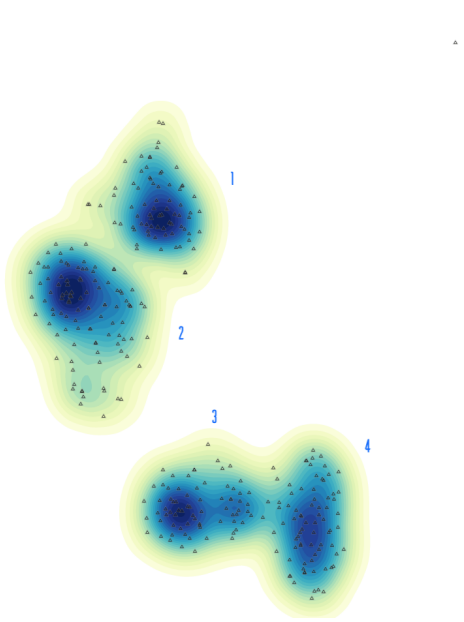
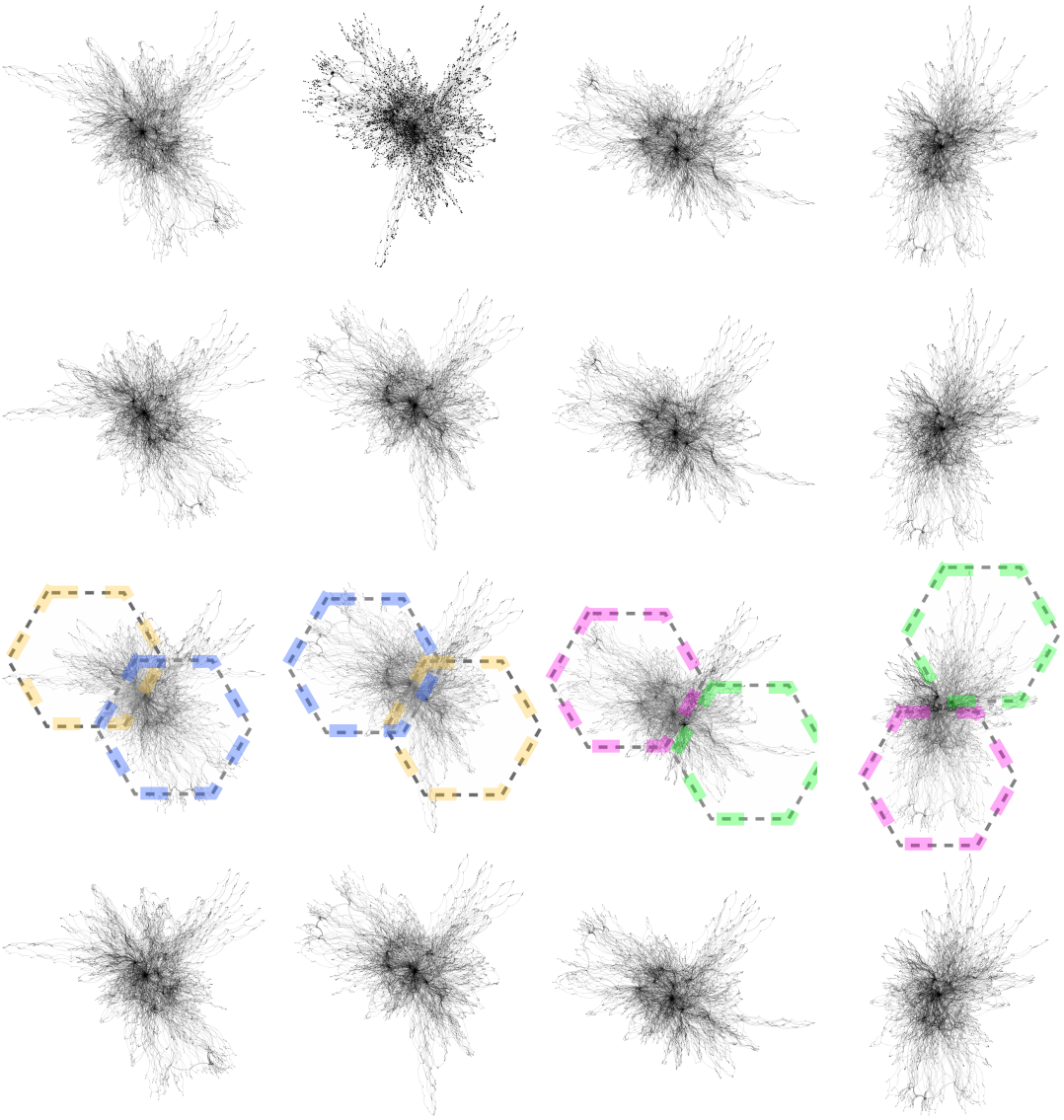


Figure 1.28: **A normalised scatter plot of 2D space produced by the t-SNE algorithm.** Each triangle represents a different arrangement of the MCM nodes shown in Figure 1.27, and the colours/density contours show the regions in which we find similar images/graphs.

Using interactivity and/or vector cluster detection techniques, it is possible to examine which files contribute to an area of high density. Figure 1.29 shows a sample of four graphs from each of the four corresponding clusters. Although individual node locations may vary, patterns on the macro scale start to emerge, with similar groups exhibiting symmetrical symmetry, e.g. groups 1/2 and 3/4. This suggests a constraint in the overall degree of freedom of the network can be attributed solely to its structure, and consequently the chemistry which forms this. The non-random nature of the produced graph layouts means that it would be possible to juxtapose a variety of mechanisms using the ForceAtlas2 layout.



(a) Group 1 (b) Group 2 (c) Group 3 (d) Group 4

Figure 1.29: **A selection of graphs for each of the labeled groubs in Figure 1.28**. These reveal that symmetric similarity between like-positioned points within the t-SNE output.

1.5.2.2 Network Branch Classification

In Subsubsection 1.5.2.1 it was seen that there exist a certain branch pattern that emerges from the structure of the MCM (Figure 1.29). Upon manual inspection of the simulations (Figure 1.27) many graphs appear to contain three branches for each graph - using this it may be hypothesized that these are a result of the mechanism, and by consequence the chemistry it describes.

To test for this, we categorise all primary emitted species into Alkanes, Alkenes, Aromatics and Terpenes. All nodes and links in close proximity are regarded as products of these species and are placed within the same group. Using a randomly selected graph from the dataset, the network is separated spatially, and nodes within the Voronoi cell (These are described in Subsubsection 1.3.1.3)

of a primary emitted species are coloured similarly.

Figure 1.30 shows a split in the MCM chemistry for the Beijing mechanism. Here it is found that it is possible to separate the MCM network into an aromatic branch, a terpene branch, an alkane and straight-chain alkene branches. Such branches not only help us identify changes of chemistry due to biogenic or anthropogenic sources but also emphasise the path taken to carbon dioxide and water. As the MCM does not contain CO_2, we see all the different groups converge on Carbon Monoxide at the centre of the figure (the white dot). Coupled with the last section, this suggests that following rotational and symmetric transformations, it is possible to compare a number of different mechanisms.

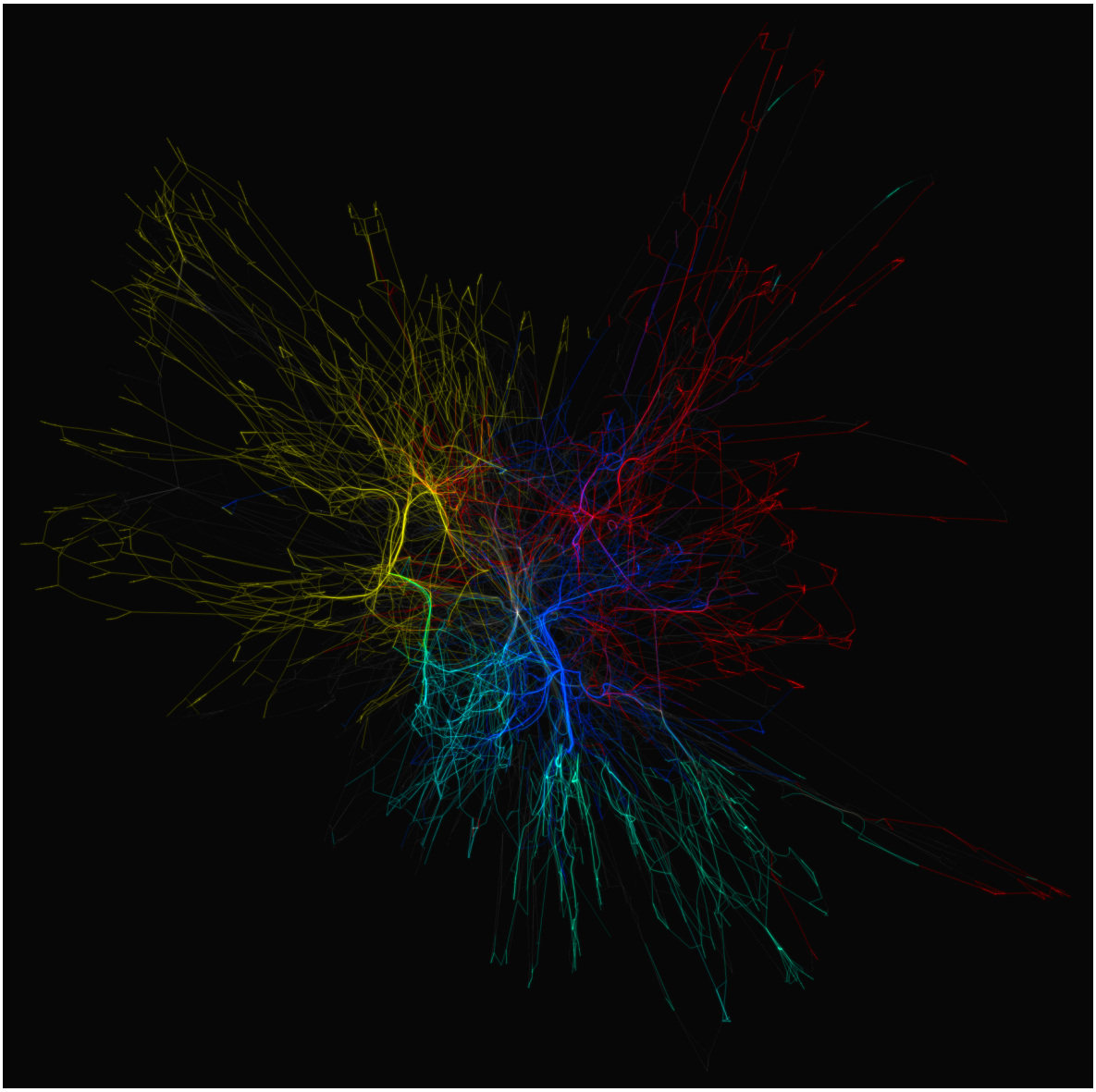


Figure 1.30: **Hi-lighting the groups of species, and their products within one of the MCM network graphs from Figure 1.27** These are Aromatics (gold) , Terpenes (turquoise) and Alkane/Alkene carbon chains (red/blue)

1.6 Summary

?? explained that the visual representation of data could make use of the pattern recognition side of the human brain. Similarly to the invention of cuneiform, we are able to use graphics to alleviate some of the cognitive strain from numerical data. It was noted that metaphor selection and storytelling are an essential part of conveying complex information to the reader and that the choice of encoding plays an integral part in this.

This chapter begins on building on those concepts of visualisation. We defined our system as a collection of relational interactions between species and chose the graph/sociograph design to represent these.

Next, we explored the different methods of graph design. These were semantic (meaning/design) and syntactic (structure). In the syntactic category, we found that the use of a force-directed graph provides a system most familiar to the reader. In addition to this, the ForceAtlas algorithm helped to produce a mathematical representation (practicality) of the relationships between the MCM network, whilst reducing the amount of clutter within the graph (visual aesthetics). In semantic design, it was noted that information about concentration, or functional groups might be represented within the network. Interactivity was found to be useful where additional information did not need to be provided but maybe enquired by at a later point in the analysis. Edge shape and design was also explored. Here a confluent graph was seen to produce the easiest to understand the structure but was the most difficult to implement. The next most useful method was edge bundling, which was used in **??**, and future work.

Although graph layouts have a range of local minima, the overall network structure of the MCM is constrained by its construction protocol (due to the allowed chemical reactions) and thus can be used to produce comparable graphs. This method of visualisation, in combination with interactive querying techniques, can aid in the comparison and understanding of large/complex chemistry simulations. This can be particularly useful in the explanation of specific interactions within a mechanism, or the exploration of temporal changes within a box-model simulation.

The next chapter builds on the use of graphs in situations where visualisation may not be possible - for example, automatically generated graphs consisting of billions of species and reactions. To do this, we apply a series of graph metrics that allow the classification and ranking of graphs, and the nodes (species) within them.

Bibliography

Archambault, D., Abello, J., Kennedy, J., Kobourov, S., Ma, K.-L., Miksch, S., Muelder, C., and Telea, A. C. (2014). *Temporal Multivariate Networks*, pages 151–174. Springer International Publishing, Cham. http://dx.doi.org/10.1007/978-3-319-06793-3_8.

Aumont, B., Szopa, S., and Madronich, S. (2005). Modelling The Evolution Of Organic Carbon During Its Gas-Phase Tropospheric Oxidation: Development Of An Explicit Model Based On A Self Generating Approach. *Atmospheric Chemistry and Physics*, 5:2497–2517. https://www.atmos-chem-phys.net/5/2497/2005/acp-5-2497-2005.pdf.

Bach, B. (2020). Confluent Graphs. https://aviz.fr/~bbach/confluentgraphs/.

Bach, B., Riche, N. H., Hurter, C., Marriott, K., and Dwyer, T. (2017). Towards Unambiguous Edge Bundling: Investigating Confluent Drawings For Network Visualization. *IEEE transactions on visualization and computer graphics*, 23(1):541–550. http://dx.doi.org/10.1109/TVCG.2016.2598958.

Baronchelli, A., Ferrer-i Cancho, R., Pastor-Satorras, R., Chater, N., and Christiansen, M. H. (2013). Networks In Cognitive Science. *Trends in cognitive sciences*, 17(7):348–360. http://dx.doi.org/10.1016/j.tics.2013.04.010.

Bastian, M., Heymann, S., and Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. *AAAI*. http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

Ben Shneiderman (1985). The eight golden rules of interface design. http://www.cs.umd.edu/~ben/goldenrules.html.

Bennett, C., Ryall, J., Spalteholz, L., and Gooch, A. (2007). The Aesthetics Of Graph Visualization. In *Computational Aesthetics in Graphics, Visualization, and Imaging*. The Eurographics Association.

Bergwerf, H. (2019). Molview. http://molview.org/.

Chernobelskiy, R., Cunningham, K. I., Goodrich, M. T., Kobourov, S. G., and Trott, L. (2012). *Force-Directed Lombardi-Style Graph Drawing*, pages 320–331. Springer Berlin Heidelberg, Berlin, Heidelberg. http://dx.doi.org/10.1007/978-3-642-25878-7_31.

Davidson, R. and Harel, D. (1996). Drawing graphs nicely using simulated annealing. *ACM Trans. Graph.*, 15(4):301–331. http://doi.acm.org/10.1145/234535.234538.

Deering, M. F. (1998). The Limits Of Human Vision. In *2nd International Immersive Projection Technology Workshop*, volume 2. https://www.swift.ac.uk/about/files/vision.pdf.

Di Battista, G., Eades, P., Tamassia, R., and Tollis, I. G. (1994). Algorithms for drawing graphs: An annotated bibliography. *Comput. Geom. Theory Appl.*, 4(5):235–282. http://dx.doi.org/10.1016/0925-7721(94)00014-X.

Di Battista, G., Mariani, F., Patrignani, M., and Pizzonia, M. (2004). *Bgplay: A System For Visualizing The Interdomain Routing Evolution*, pages 295–306. Springer Berlin Heidelberg, Berlin, Heidelberg. http://dx.doi.org/10.1007/978-3-540-24595-7_27.

Dianati, N. (2016). Unwinding the hairball graph: Pruning algorithms for weighted complex networks. *Phys. Rev. E*, 93:012304. https://link.aps.org/doi/10.1103/PhysRevE.93.012304.

Dick Derwent, Andrea Fraser, J. A. M. J. (2010). Evaluating The Performance Of Air Quality Models. https://uk-air.defra.gov.uk/assets/documents/reports/cat05/1006241607_100608_MIP_Final_Version.pdf.

Dixon, D. (2012). *Analysis Tool Or Research Methodology: Is There An Epistemology For Patterns?*, pages 191–209. Palgrave Macmillan UK, London. http://dx.doi.org/10.1057/9780230371934_11.

Duyckaerts, C. and Godefroy, G. (2000). Voronoi tessellation to study the numerical density and the spatial distribution of neurones. *Journal of Chemical Neuroanatomy*, 20(1):83 – 92. http://www.sciencedirect.com/science/article/pii/S0891061800000648.

Dwyer, T., Koren, Y., and Marriott, K. (2006a). Drawing directed graphs using quadratic programming. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):536–548.

Dwyer, T., Koren, Y., and Marriott, K. (2006b). Ipsep-Cola: An incremental procedure for separation constraint layout of graphs. *IEEE Trans. Vis. Comput. Graph.*, 12(5):821–828.

Dwyer, T., Marriott, K., and Stuckey, P. J. (2006c). *Fast Node Overlap Removal*, pages 153–164. Springer Berlin Heidelberg, Berlin, Heidelberg. http://dx.doi.org/10.1007/11618058_15.

Eades, P. (1984). A heuristic for graph drawing. pages 149–160. cited By 1.

Ellis, D. (2018). Animation Of The Evolution Of Chemistry Graph Of Beijing. https://github.com/wolfiex/DanEllisThesis/blob/master/daynight_26mb.gif.

Emmerson, K. M. and Evans, M. J. (2009). Comparison of tropospheric gas-phase chemistry schemes for use within global models. *Atmospheric Chemistry and Physics*, 9(5):1831–1845. https://www.atmos-chem-phys.net/9/1831/2009/.

Foo, B. (2019). Memory Underground - Convert Your Memories Into A Subway Map - Home. http://memoryunderground.com/.

Friedrich, C. and Schreiber, F. (2004). Flexible layering in hierarchical drawings with nodes of arbitrary size. In *Proceedings of the 27th Australasian Conference on Computer Science - Volume 26*, ACSC '04, pages 369–376, Darlinghurst, Australia, Australia. Australian Computer Society, Inc. http://dl.acm.org/citation.cfm?id=979922.979966.

Fruchterman, T. M. J. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164. https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380211102.

Förster, H., Ganian, R., Klute, F., and Nöllenburg, M. (2019). On strict (outer-)confluent graphs.

García-Pérez, G., Allard, A., Ángeles Serrano, M., and Boguñá, M. (2019). Mercator: Uncovering Faithful Hyperbolic Embeddings Of Complex Networks. *arxiv*. http://arxiv.org/abs/1904.10814.

Gershon, N. and Page, W. (2001). What storytelling can do for information visualization. *Commun. ACM*, 44(8):31–37. http://doi.acm.org/10.1145/381641.381653.

Goodrich, M. T. and Wagner, C. G. (1998). *A Framework For Drawing Planar Graphs With Curves And Polylines*, pages 153–166. Springer Berlin Heidelberg, Berlin, Heidelberg. http://dx.doi.org/10.1007/3-540-37623-2_12.

Görg, C., Pohl, M., Qeli, E., Xu, K., Ebert, A., and Meyer, J. (2007). *Visual Representations*, pages 163–230. Springer Berlin Heidelberg, Berlin, Heidelberg. http://dx.doi.org/10.1007/978-3-540-71949-6_4.

Harari, Y. (2015). *Sapiens: A Brief History Of Humankind*. Harper. https://books.google.co.uk/books?id=FmyBAwAAQBAJ.

Hazewinkel, M. (1997). *Encyclopaedia Of Mathematics: Supplement*. Number v. 1 in Encyclopaedia of Mathematics. Springer Netherlands. https://books.google.co.uk/books?id=3ndQH4mTzWQC.

Holten, D. (2006). Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748.

Hu, Y. (2004). Efficient, High-Quality Force-Directed Graph Drawing. *web*. http://yifanhu.net/PUB/graph_draw.pdf.

Jacomy, M., Venturini, T., Heymann, S., and Bastian, M. (2014). Forceatlas2, A Continuous Graph Layout Algorithm For Handy Network Visualization Designed For The Gephi Software. *PloS one*, 9(6):e98679. http://dx.doi.org/10.1371/journal.pone.0098679.

Jankun-Kelly, T. J., Dwyer, T., Holten, D., Hurter, C., Nöllenburg, M., Weaver, C., and Xu, K. (2014). *Scalability Considerations For Multivariate Graph Visualization*, pages 207–235. Springer International Publishing, Cham. http://dx.doi.org/10.1007/978-3-319-06793-3_10.

Jenkin, M., Watson, L., Utembe, S., and Shallcross, D. (2008). A common representative intermediates (cri) mechanism for voc degradation. part 1: Gas phase mechanism development. *Atmospheric Environment*, 42(31):7185 – 7195. http://www.sciencedirect.com/science/article/pii/S1352231008006742.

Jenkin, M. E., Saunders, S. M., and Pilling, M. J. (1997). The Tropospheric Degradation Of Volatile Organic Compounds: A Protocol For Mechanism Development. *Atmospheric environment*, 31(1):81–104. http://www.sciencedirect.com/science/article/pii/S1352231096001057.

Johnson, S. (2010). *Where Good Ideas Come From.* Penguin Publishing Group. https://books.google.co.uk/books?id=3H2Xg5qxz-8C.

Kerren, A., Purchase, H. C., and Ward, M. O. (2014). *Introduction To Multivariate Network Visualization*, pages 1–9. Springer International Publishing, Cham. http://dx.doi.org/10.1007/978-3-319-06793-3_1.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. http://science.sciencemag.org/content/220/4598/671.

Klicpera, J., Bojchevski, A., and Günnemann, S. (2018). Predict Then Propagate: Graph Neural Networks Meet Personalized Pagerank. *Arxiv*. http://arxiv.org/abs/1810.05997.

Kohlbacher, O., Schreiber, F., and Ward, M. O. (2014). *Multivariate Networks In The Life Sciences*, pages 61–73. Springer International Publishing, Cham. http://dx.doi.org/10.1007/978-3-319-06793-3_4.

Kumar, G. and Garland, M. (2006). Visual exploration of complex time-varying graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):805–812.

Lu, S. (2019). D3-Annotate. https://d3-annotation.susielu.com/.

Lyons, K. A. (1992). Cluster busting in anchored graph drawing. In *Proceedings of the 1992 Conference of the Centre for Advanced Studies on Collaborative Research - Volume 1*, CASCON '92, pages 7–17. IBM Press. http://dl.acm.org/citation.cfm?id=962198.962200.

Ma, K. and Muelder, C. W. (2013). Large-scale graph visualization and analytics. *Computer*, 46(7):39–46.

Maaten, L. v. d. and Hinton, G. (2008a). Visualizing Data Using T-Sne. *Journal of machine learning research: JMLR*, 9(Nov):2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Maaten, L. v. d. and Hinton, G. (2008b). Visualizing Data Using T-Sne. *Journal of machine learning research: JMLR*, 9(Nov):2579–2605. http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

Martin, S., Brown, W., Klavans, R., and Boyack, K. (2011). Openord: An open-source toolbox for large graph layout. *Proc SPIE*, 7868:786806.

Martin Grandjean (2016). Connected World: Untangling The Air Traffic Network. http://www.martingrandjean.ch/connected-world-air-traffic-network/.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092. http://scitation.aip.org/content/aip/journal/jcp/21/6/10.1063/1.1699114.

Michal, G. (1965). Metabolic Pathways. https://www.roche.com/sustainability/philanthropy/science_education/pathways/pathways-ordering.htm.

Montañez, A. (2016). How Science Visualization Can Help Save The World. https://blogs.scientificamerican.com/sa-visual/how-science-visualization-can-help-save-the-world/.

Mortenson, M. (1999). *Mathematics For Computer Graphics Applications*. Industrial Press. https://books.google.co.uk/books?id=YmQy799flPkC.

Muelder, C., Gou, L., Ma, K.-L., and Zhou, M. X. (2014). *Multivariate Social Network Visual Analytics*, pages 37–59. Springer International Publishing, Cham. http://dx.doi.org/10.1007/978-3-319-06793-3_3.

Needham, M. and Hodler, A. E. (2019). Practical Examples In Apache Spark & Neo4J. *O'Reilly*. https://neo4j.com/neoassets/graphbooks/Graph_Algorithms_Neo4j.pdf.

Noack, A. (2004). *An Energy Model For Visual Graph Clustering*, pages 425–436. Springer Berlin Heidelberg, Berlin, Heidelberg. http://dx.doi.org/10.1007/978-3-540-24595-7_40.

Norman, D. (2005). *Emotional Design: Why We Love (Or Hate) Everyday Things*. Basic Books. https://books.google.nl/books?id=h_wAbnGlOC4C.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot,

M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pohl, M., Schmitt, M., and Diehl, S. (2009). Comparing The Readability Of Graph Layouts Using Eyetracking And Task-Oriented Analysis. In *Computational Aesthetics in Graphics, Visualization, and Imaging*. The Eurographics Association.

Purchase, H. (1997). *Which Aesthetic Has The Greatest Effect On Human Understanding?*, pages 248–261. Springer Berlin Heidelberg, Berlin, Heidelberg. http://dx.doi.org/10.1007/3-540-63938-1_67.

Purchase, H. C. (2002). Metrics for graph drawing aesthetics. *Journal of Visual Languages and Computing*, 13(5):501 – 516. http://www.sciencedirect.com/science/article/pii/S1045926X02902326.

Purchase, H. C., Colpoys, L., Carrington, D., and McGill, M. (2003). *Uml Class Diagrams: An Empirical Study Of Comprehension*, pages 149–178. Springer US, Boston, MA. http://dx.doi.org/10.1007/978-1-4615-0457-3_6.

Roberts, J. C., Yang, J., Kohlbacher, O., Ward, M. O., and Zhou, M. X. (2014). *Novel Visual Metaphors For Multivariate Networks*, pages 127–150. Springer International Publishing, Cham. http://dx.doi.org/10.1007/978-3-319-06793-3_7.

Sangers, A., van Heesch, M., Attema, T., Veugen, T., Wiggerman, M., Veldsink, J., Bloemen, O., and Worm, D. (2019). Secure Multiparty Pagerank Algorithm For Collaborative Fraud Detection. In *Financial Cryptography and Data Security*, pages 605–623. Springer International Publishing. http://dx.doi.org/10.1007/978-3-030-32101-7_35.

Sangkloy, P., Burnell, N., Ham, C., and Hays, J. (2016). The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Trans. Graph.*, 35(4). https://doi.org/10.1145/2897824.2925954.

Schreiber, F., Kerren, A., Börner, K., Hagen, H., and Zeckzer, D. (2014). *Heterogeneous Networks On Multiple Levels*, pages 175–206. Springer International Publishing, Cham. http://dx.doi.org/10.1007/978-3-319-06793-3_9.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–, Washington, DC, USA. IEEE Computer Society. http://dl.acm.org/citation.cfm?id=832277.834354.

Shneiderman, B. (1997). *Designing The User Interface: Strategies For Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition.

Staples, J., Nickerson, D. A., and Below, J. E. (2013). Utilizing graph theory to select the largest set of unrelated individuals for genetic analysis. *Genetic Epidemiology*, 37(2):136–141. https://onlinelibrary.wiley.com/doi/abs/10.1002/gepi.21684.

Stefaner, M. (2020). Truth & Beauty - Multiplicity. https://truth-and-beauty.net/projects/multiplicity.

Steven Franconeri (2018). Openvis Conference Proceedings. https://www.youtube.com/watch?v=Jq2Rc0WlYTE.

Taylor, M. and Rodgers, P. (2005). Applying graphical design techniques to graph visualisation. In *Ninth International Conference on Information Visualisation, 06-08 July 2005, London, England: Proceedings*, pages 651–656. IEEE Computer Society. http://kar.kent.ac.uk/14297/.

Thomas, P. (1952). *Conformal Projections In Geodesy And Cartography*. Special publication. Coast and Geodetic Survey. https://books.google.co.uk/books?id=7a60MQEACAAJ.

VTL (2019). Visual Thinking Lab. http://visualthinking.psych.northwestern.edu/.

Ware, C. (2013). Chapter two - the environment, optics, resolution, and the display. In Ware, C., editor, *Information Visualization (Third Edition)*, Interactive Technologies, pages 31 – 68. Morgan Kaufmann, Boston, third edition edition. http://www.sciencedirect.com/science/article/pii/B9780123814647000028.

Ware, C., Purchase, H., Colpoys, L., and McGill, M. (2002). Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110. http://dx.doi.org/10.1057/palgrave.ivs.9500013.

Wybrow, M., Elmqvist, N., Fekete, J.-D., von Landesberger, T., van Wijk, J. J., and Zimmer, B. (2014). *Interaction In The Visualization Of Multivariate Networks*, pages 97–125. Springer International Publishing, Cham. http://dx.doi.org/10.1007/978-3-319-06793-3_6.

Chapter 2

Computational Learning, Visualisation and Clustering:

Learning species structure using unsupervised machine learning.

"So, in the interests of survival, they trained themselves to be agreeing machines instead of thinking machines. All their minds had to do was to discover what other people were thinking, and then they thought that, too."

- Kurt Vonnegut, *Breakfast of Champions*

2.1 Introduction

Historical Significance

The established process of trial and error has always underpinned our survival [Noble, 1957]. Babies are born to rely on a set of sensory reflexes and a framework for physical reasoning [Baillargeon and Carey, 2012], and with these, we develop methods to navigate the influence of change within a physical, and auditory space [Lynch, 2011]. This method of decision making is reflected in our adult lives with ideas and actions being limited in choice by our intuition and experience [Descartes and Lafleur, 1960]. In science, we apply a methodological framework consisting of a continuous assessment of scepticism, educated guessing (hypothesising) and rigorous practical testing. Specialists accrue years of practical and theoretical knowledge within a narrow field and can identify areas of potential gain and futility. Yet even with all prior experience, the discovery of new and untested techniques involve the tortuous traipsing through a sea of uncertainty. Such a methods sometimes prove fruitful, through accidental discoveries of items such as x-rays, penicillin... [Roberts, 1989]; finding novel applications for existing methods such as optical tweezers for chemistry or the abstract field of maths utilised by Einstein [REF], but more often than not end in the constant evolution of a pre-existing project with no apparent result.

Theory And Simulation In Science

Until recently much of the experimentation possible was limited by resources, levels of knowledge available technology. With the increase of computation power, we have been able to not only increase our understanding but also run theoretical simulations to guide exploratory efforts with an impact on real-world applications [Oliveira et al., 2006; T. Leube et al., 2018; Morozov, 2016; Yu-ChenLo, 2018]. However, as our ability to record and produce data increases, the need for the scientific method diminishes [Anderson, 2008]. Here the application of ‘big data’ tools and algorithms can provide insights and correlations much more compelling than the predictive capabilities of constantly changing models - “Since all models are wrong the scientist cannot obtain a "correct" one by excessive elaboration” - Box [1976]. As our level of attainable technology increases, so does the complexity of the data collected. New datasets tend to be large, complex and highly multivariate. Although this dramatically improves the quality of science, the difficulty lies in trying to represent it in such a way that we may successfully access the reliability of the results. Since simple bar and line graphs are no longer applicable, one solution falls within a class of unsupervised machine learning techniques called dimensionality reduction (DR).

Chapter Aims

In **??**, we looked at visual representation as a way of understanding complex systems. **??** showed that the chemical properties could be inferred (visually) from the node-link graph structure of a mechanism. Similarly, Chapter 1 and **??** located the presence of important species and clusters of similar properties by applying mathematical algorithms to the graph network. As opposed to attempting to visualise complex data, this chapter looks at learning the structure of a chemical species and simplifying it into two dimensions. Here it is possible to extract key features of like-groups through the use of vector clustering, which unlike the graph clustering in **??** works by determining the density between points on a plane.

The chapter begins with the introduction of the chemical system, and the various methods for representing species structure within it (Section 2.2). Next, we define the dimensionality reduction methods, which are to be used to simplify the inputs above (**??**). This is followed by a brief overview of the visualisation methodology (**??**). Finally, all three sections are combined to produce a set of result and conclusions about the use of DR to identify species structure.

2.2 Species Of The Mcm And Ways To Represent Them.

The master chemical system (as defined in all previous chapters), represents our foremost knowledge of gas-phase chemistry within the troposphere. **??** shows that information about a species structure is encoded within its reactions, much of which can be attributed to the well-defined construction protocols.

This section explores the different methods of representing a species structure, intending to provide a machine built algorithm with the highest amount of information about each species and its functionality. A range of input types will be evaluated against several dimensionality reduction algorithms to isolate which chemical properties are most 'picked up'.

2.2.1 Input Generation

The MCM provides species information in the form of a species 'smiles' (Subsubsection 2.2.3.2) and the IUPAC InChi string [Heller et al., 2013]. Within this chapter, we use only the smiles string, which is either manually processed using regular expressions or with the aid of pythons RDKIT package [Landrum et al., 2019]. There are seven different methods for representing the chemistry; these are outlined below.

2.2.2 Manual Categorisation

Reactions within the MCM are determined by a set of rules (PROTOCOL SECTION). These mimic the process a chemist my discover new species and often rely on the bond availability and functionalisation of a species. Since the present functional groups are the benchmark of whether a DR algorithm has successfully separated species structure, it makes sense to run a unit test using the known functional groups of a species as the input.

To generate the functional groups the regular expressions in Table 2.1 are used[1] on the smiles strings (described in Subsubsection 2.2.3.2) for each species. In extracting the functional groups, we can plot the likeliness a species with a certain group is likely to have another using a chord diagram - Figure 2.1. Since most species contain a multitude of functional groups, the separation of these into 'tidy' clustered groups seems unlikely.

PAN	`C\\(=O\\)OON\\(=O\\)=O$|^\\[O-{0,1}\\]\\[N\\+{0,1}\\]\\(=O\\)OOC|O=N\\(=O\\)OOC\\(=O\\)|C\\(=O\\)OO\\[N\\+{0,1}\\]\\(=O\\)\\[O-{0,1}\\]`
Carb. Acid	`[^O](C\\(=O\\)O$|^OC\\(=O\\))`
Ester	`[\^O](C\(=O\)O\b|OC\(=O\))C`
Ether	`(\([\^O=]+\))*C(\([\^O=]+\))*O(\([\^O=]+\))*C(\([\^O=]+\))*`
Per. Acid	`c\\(=O\\)OO$|^OO\\(=O\\)C`
Nitrate	`O(NO2\b|NOO\b|N\(=O\)=O|\[N\+\](?:\[O-\]|\(=O\)){2})`
Aldehyde	`C=O$|^O=C`
Ketone	`C\(=O\)C`
Alcohol	`CO\\b|(?=^\\b)(?!^\\[)CO.|(?=^\\b)(?!^\\[)OC.|\\(O\\)|C\\)O(\\b|[^O]`
Criegee	`\[O-\]\[O\+\]`
Alkoxy rad	`\[[\/]{0,1}CH{0,1}\]|\b[\^O]\[O\.{0,1}\]`
Peroxyacyl rad	`\\ w\(=O\)O\[O\.{0,1}\]`

Table 2.1: CHECKKKKKKK!!!!!!!!! A set of regular expressions that may be used to determine the number of occurrences of a functional group within a SMILES string.

[1] To see the structure of each functional group type, go to **??**.

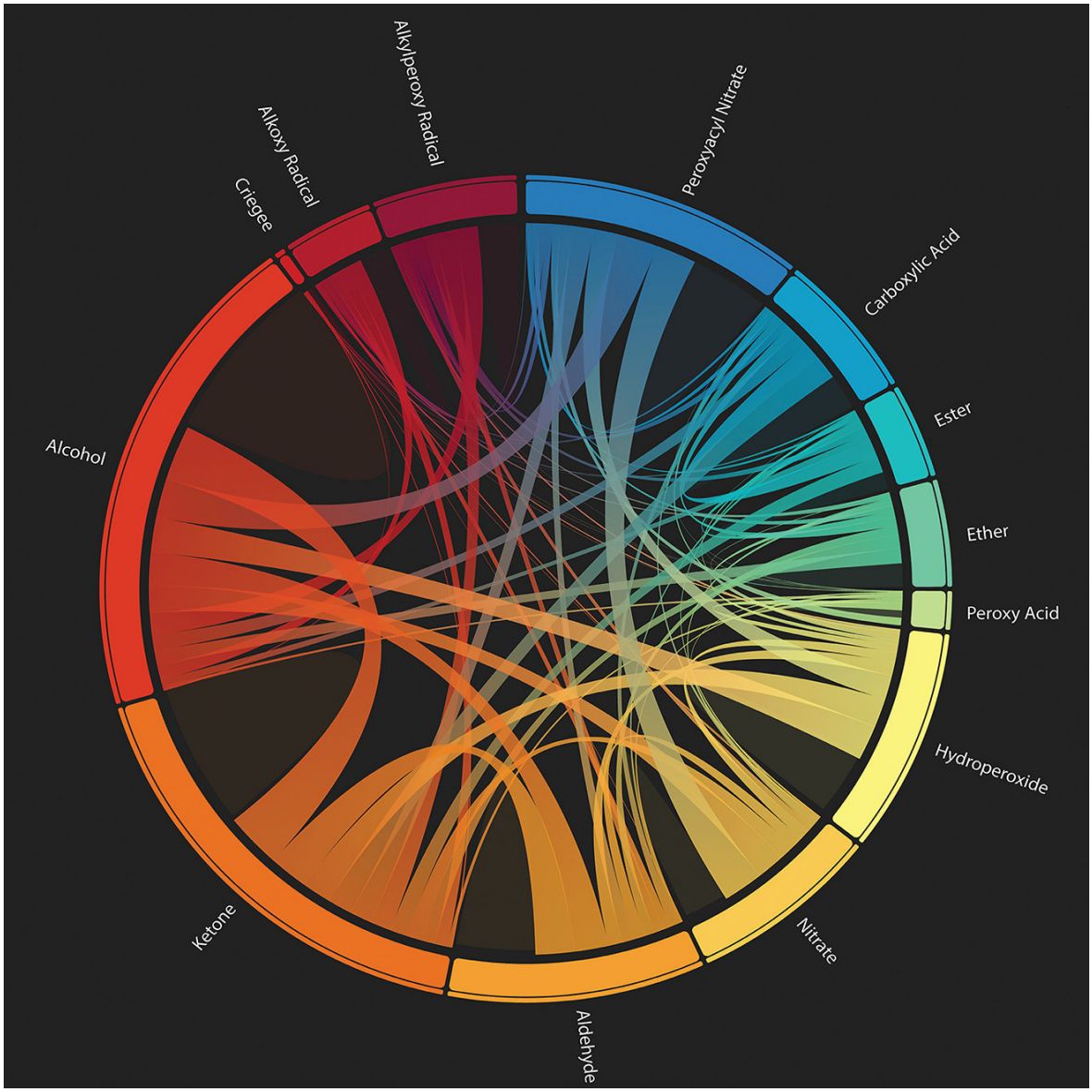


Figure 2.1: **The multifunctionality of the MCM.** A chord diagram showing the functionalisation of a species within the MCM. Arc sizes represent what percentage of all functional groups in the MCM mechanism a group contains. Translucent areas of no outwards links represent species with multiples of a certain functional group, of which Alcohols and Ketones have the most. Source: [Ellis, 2019]

2.2.3 Tokenization

As computer algorithms are unable to understand words or their meaning, we have to first categorise the data into groups. Tokenisation is the conversion of a string into characters and representing them with a numerical equivalent. In doing so, a string of characters can be converted into a numerical vector, allowing for its representation in a latent vector space. Within our input selection, we have two sets of inputs we can convert. These are the species names, and their smiles string representation.

2.2.3.1 Species Names

In **??** it was shown that the dedicated species names for species in the CRI mechanism were often representative of their structural properties. This adage also applies for the MCM, where an intuitive naming convention is used. This is often derived as part of the construction protocol, where a species names reflect its own, or its precursor's structure (which it will have at least in-part inherited).

Although this is not the most robust method of defining the structure, it allows for a straightforward test of the algorithms, for which the user can quickly compare the human-readable output.

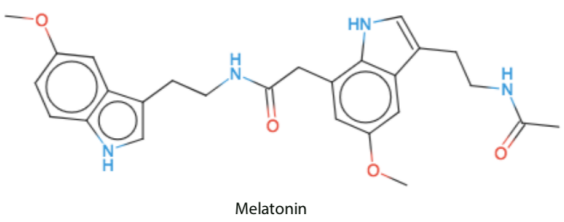
2.2.3.2 Smiles Strings

Smiles ('Simplified Molecular-Input Line-Entry System') provide a human-readable representation of the molecular structure, [Weininger, 1988]. They offer a linear human-readable description of the chemical composition within a molecule - making it easy to visually check the construction of a species without any additional work. Besides, their role in generating the molecular fingerprints in Subsection 2.2.5 makes it a useful comparison to make when evaluating methods of structure representation.

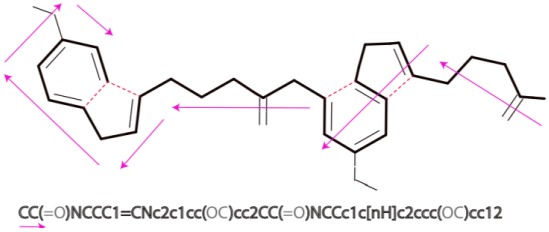
Construction Methodology of SMILES strings

The construction of a SMILES string happens in three parts:

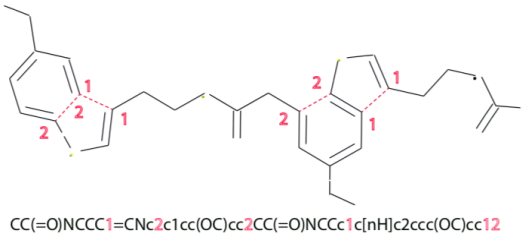
1. The smiles string is built by creating the longest possible chain to form a molecule backbone. Figure 2.2b
2. This may within itself contain aromatic rings denoted by the lowercase carbons and a number corresponding to the location of each break cycle. Figure 2.2c
3. Finally all the functional groups and branches attached to the main backbone are added. These are nested within the parenthesis to show that they are not part of the skeletal backbone. Figure 2.2d



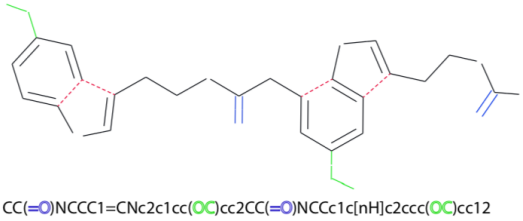
(a) Structure of Melatonin



(b) Step 1 : Building the C chain backbone.



(c) Step 2 : Aromatic Rings



(d) Step 3 : Functional Groups

Figure 2.2: **Construction process of a smiles string.** The example compound is Melatonin. Although this does not exist within the atmosphere, it provides a clear example of the smiles string methodology. Figure 2.2a is made using smiles drawer: [Probst and Reymond, 2018]

2.2.4 Graph Inspired

?? - **??** have shown the role of graphs in revealing network properties and structure. Graphs in themselves can simplify relational data into two/three dimensions for visualisation and algorithmic clustering. Continuing this trend, we can represent a species structure in the form of a graph (Subsubsection 2.2.4.1), as well as converting the structure of a mechanism for dimensionality reduction (Subsubsection 2.2.4.2)

2.2.4.1 The Species Graph (Fingerprint)

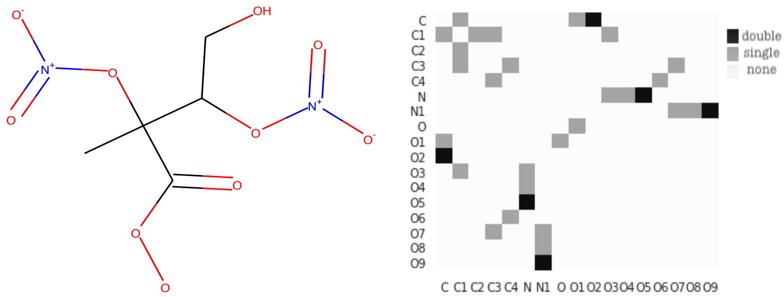
The structure of a species has long represented using a graph-like layout, **??**. It, therefore, follows that other methods for representing the graph structure would also apply. One such way is the use of an adjacency (or relational) matrix to describe the relationships between atoms and bonds in a species. Such a methodology is already used in the construction of bond and z-matrixes [Aumont et al., 2005; Parsons et al., 2005].

The construction of a structure matrix/graph begins with a chemical species. Here the relationships between atoms (Figure 2.3b) is converted into an adjacency matrix (Figure 2.3c). However, since species have different numbers of each atom, a template allowing us to compare different graphs is required. To do this a maximum occurrence table (Figure 2.3a) is created. Here, for example, BCARY $C_{15}H_{24}$, a sesquiterpene contains the most carbon atoms of any species within the MCM. This universal matrix is now able to contain any possible combination of atoms in a species.

As machine learning algorithms only vectors as an input, it is possible to decompose the 37^2 element adjacency matrix into rows, which can then be joined together, Using this method we create a one-dimensional array (vector) of 259 elements (518 bytes) to represent our species.

Atom	**Max**
C	15
Cl	4
O	12
N	3
S	1
BR	2

(a)



(b) (c)

Figure 2.3: **Constructing a graph from species structure.** (a) shows the maximum number of times an atom occurs for any single species in the MCM. (b) depicts the graph-like chemical structure of INB_1NBCO_3. This is a highly processed species stemming from Isoprene, and this makes for a good example of the bond matrix. Finally, a matrix representing the bonds in INB_1NBCO_3 is created from the maximum possible occurrence matrix from (a). For simplicity, empty row/column pairs have been removed to produce (c). This matrix will always be symmetrical as the bonds do not have a direction.

2.2.4.2 Node Embeddings (Node2Vec)

?? and Chapter 1 showed that the underlying structure of a chemistry mechanism graph contains information about the species and reactions within it. In Figure 2.4 colour represents the ratio of potential oxidation of a species. Here as emitted species become progressively more processed, the number of bonds which may be oxidised diminishes (lighter colours near the centre) until they eventually form carbon dioxide and water.

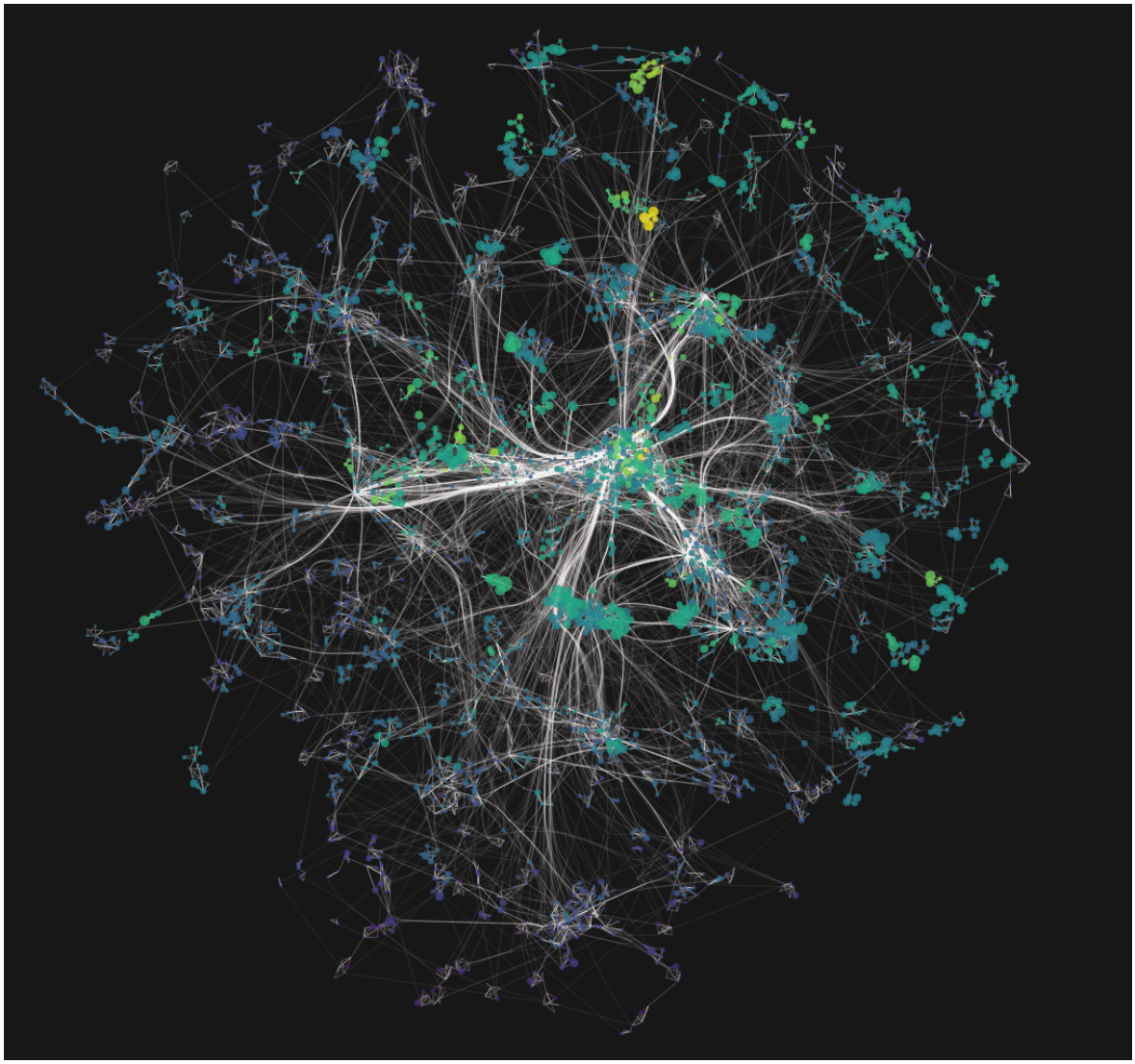


Figure 2.4: **The graph of an MCM subset representing the chemistry within Beijing.** Here colours show the increase of O−C ratio as species are oxidised (lighter). All emitted species ultimately tend towards carbon monoxide which is at the centre of the graph.

This type of structural information can be extracted through the use of a natural language processing package capable of transforming a graph into a vector - node2vec [Grover and Leskovec, 2019]. Since this may also be used for dimensionality reduction, it is described within the next section (Subsection 2.3.6).

2.2.5 Molecular Fingerprints

In the field of chemical informatics, molecular fingerprints (or structural keys) are used to encode and query structural properties of species. Their binary representation makes them suitable for dimensionality reduction and the exploration of fo chemical space (a type of property space constructed using pre-determined features and boundary conditions).

Here species properties are often split into structural and psyico-chemical groups - which has used such as the discovery of natural analogues (which circumvent problems such as intolerances in medicine [Spahn et al., 2017]). Although there exist many different types of molecular fingerprints, the two main ones that will be explored are molecular quantum numbers (MQN) and the molecular access system (MACCS).

2.2.5.1 Molecular Quantum Numbers (Mqn)

In chemistry the shape, phase and electron occupancy of an atom may be described through the use of four quantum numbers: the n principle quantum number, I angular momentum quantum number, M_i magnetic quantum number and M_s spin quantum number. The rationalisation of elements based on their structure, and by consequence reactivity, has led to the most iconic tool of the modern-day chemist - the periodic table, where increasing atomic numbers follow the principal quantum number [Wang and Schwarz, 2009]. In representing a molecule as a set of 42 quantum numbers, MQN fingerprints produce a multi-dimensional mapping of atom, bond, polarity and topology count [Nguyen et al., 2009].

2.2.5.2 Molecular Access System (Maccs)

MACCS keys are a 164[2] bit structural keys formulated through answering a series of structure-related questions. Developed by MDL Information Systems [, MDL], their main purpose lies in being a SMILES Arbitrary Target Specification (SMARTS) system for substructure searching. However, their distinct structure key format makes them highly suitable for similarity detection. In many cases, the optimised version of MACCS keys is cited ([Durant et al., 2002]), although most use cases exploit a variation of the undocumented 166bit keys. We use the implementation presented by [Landrum et al., 2019; rdkit, 2019] for all molecular fingerprints in this section.

2.3 Dimensionality Reduction Methods

In the last section, we described several methods in which the chemical structure of a species could be encoded for direct comparison. However, since each input consists of a multitude of elements, it is still not a simple task to determine the differences and similarity between all species in mechanisms. Dimensionality reduction is the process of reducing the number of random variables and only presented a set of principal values, by mapping a high-dimensional space into a low-dimensional one [Roweis

[2]They are 166-bit keys, although there is no real agreement to what the 44th keys' purpose is, and therefore it is often omitted. Within RDKIT this is denoted by a ? [rdkit, 2019].

and Saul, 2000]. This allows us to flatten a multivariate input into the two dimensions required for a simple scatter plot.

In this section, we begin by explaining the data preparation required for dimensionality reduction (**??**) before describing the different possible methods of reducing the dimensions of a dataset.

2.3.1 Preperation Of The Data

Real-world data is rarely preformatted in such a way that it can be used directly within a computational model. Often values need to be cleaned and corrected to be fit for purpose. In the interest of completeness, the two main methods of data adjustment for machine learning are outlined below. These are normalisation and standardisation.

Normalisation

In the data is without (dimensionless) or of a single unit, it is possible to rescale the data between a range - most commonly 0,1. In doing so it is possible to interpret the importance of value in contrast to the largest recorded value. This gives us a percentage scale spanning the range of the data. Such a range is useful in the definition of colourmaps and describing the importance of value relative to the dataset. To rescale a dataset we shift the minimum value to zero, then divide by the new maximum of the dataset (Note this is equivalent to the range of the unshifted dataset.)

$$n(x_i) = \frac{x_i - \min_x}{\max_x - \min_x} \tag{2.1}$$

Standardisation

If the components we wish to compare are of different units or are expressed with a different scale, normalising them would not produce meaningful data. Instead, it is possible to standardise the data by looking at each points deviation from the mean. Here the variation of the mean for a dataset is divided by the standard deviation to produce a value between {-1,1}, Equation 2.2. In statistics this is known as the 'z-score'[3]

$$z(x_i) = \frac{x_i - \mu_x}{S} \tag{2.2}$$

[3]Possibly because of the American spelling of standardi**Z**ation?

2.3.2 Principle Component Analysis

One of the most well-known dimensionality reduction methods is the determination of the principal components through the use of Principal Component Analysis (PCA). PCA increases the readability of a dataset by creating a set of new uncorrelated variables which maximise the variance [Jolliffe and Cadima, 2016].

PCA works on the assumption that components within a dataset are linear combinations of each other. By simplifying these linear combinations, it is possible to identify the elements which explain the most variability in a dataset - these are the principal components.

A more straightforward interpretation of this would be to adjust the direction of each axis of the data, such that its projection has the most prominent variability. In doing so, it is possible to determine which components contribute the most to changes in the dataset [F.R.S., 1901; Hotelling, 1933]. An example of this is seen in Figure 2.5, where the second component of the original data can be removed with little effect on the overall result of the data. Such methods have applications in compression and signal filtering [Hernandez and Mendez, 2018; Hamadache and Lee, 2017].

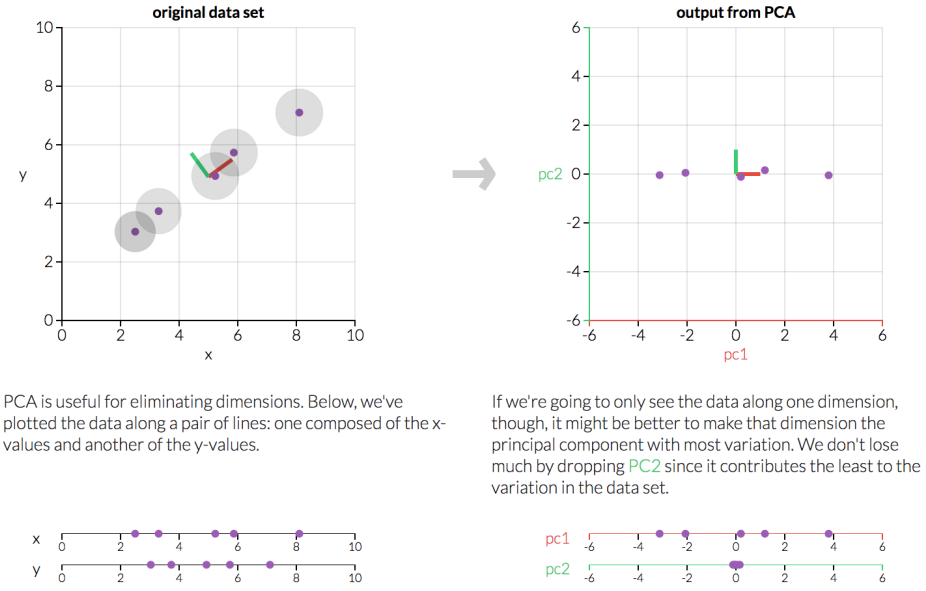


Figure 2.5: **Determining the Principal Compnent of a sample dataset.** It can be seen that in a change in axis to follow the first principal component (right), it is possible to explain most of the variation in the samle dataset (left). Source: [Powell, 2020]

2.3.2.1 Mathematical Explanation Of Pca

Note: *The basic statistics/mathematics required to understand this section is shown in* **??**. *Please read this if you are not familiar with any of the terms below.*

The mathematics behind PCA consists of first calculating the covariance matrix - an $n \times n$ matrix

outlining how strongly each variable changes with every other. Using this we can calculate both the eigenvalues and eigenvectors of the matrix [4]. This can be done using a computational package such as numpy or scipy [Oliphant, 2006; Jones et al., 01].

We can now sort the eigenvector columns by influence using their eigenvalues—this way a feature dataset can be produced by removing vectors of low importance. The final feature dataset can now be transposed and multiplied by the transpose of the original dataset. This results in an output dataset containing each principal component of the desired dimension.

2.3.3 T-Distributed Stochastic Neighbor Embedding (T-Sne)

t-SNE is an algorithm designed with visualisation in mind [Maaten and Hinton, 2008]. Rather than representing the data through a series of linear transformations, t-SNE uses local relationships to create a low-dimensional mapping, much in the same way as a fully connected force graph, Figure 2.6. This allows the ability to capture non-linear structures in the data which cannot be accomplished through linear mapping methods (e.g. PCA).

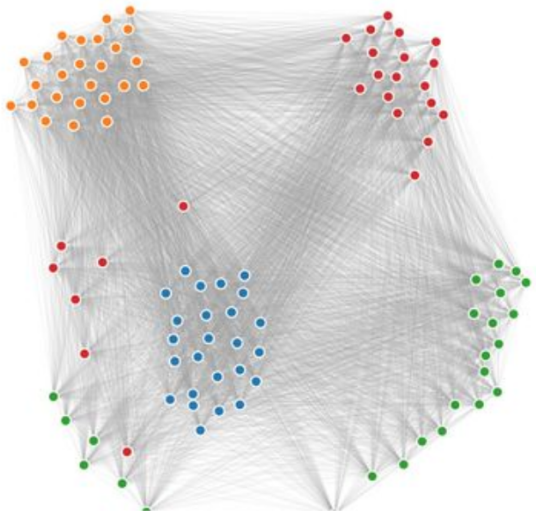


Figure 2.6: **Representing the t-SNE algorithm as a fully connected force graph.** Here each node is attached to every other node. Nodes with a strong relationship are pulled closer together than those with a weaker one.

The algorithm itself can be simplified into two parts,

1. Create a probability distribution which dictates relationships between neighbouring points
2. Recreate a lower-dimensional space following the probability distribution established in 1.

[4] These need to be unit vectors, although most packages already do this out of the box.

and is described in Subsubsection 2.3.3.1.The main reason t-SNE produces good results is that it can handle the **'crowding problem'** very well. The crowding problem is a product of the 'curse of dimensionality'. In a high dimensional space, the surface of a sphere will grow much quicker than one in a lower dimension space. This means that the higher dimension spaces will have more points at a medium distance from a certain point, Figure 2.7. When we map our data into a lower dimension, data will try to gather at its medium distance, resulting in a more 'squashed', and thus crowded, output.

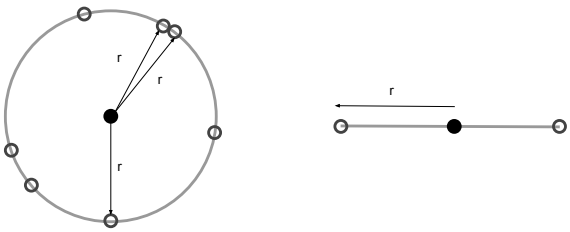


Figure 2.7: **An example of how the curse of dimensionality affects the mapping of points a certian distance from eachother.**

2.3.3.1 Mathematical Explanation Of T-Sne

In the original paper [Maaten and Hinton, 2008], the algorithm is described using the etymologic dissection of its name.

Step 1

First we begin with Stochastic Neigbour Embedding (SNE) - the distribution across neignbouring datapoints in our high dimension space. This is done by converting the high dimensional Euclidian distances between points into conditional probabilities representing their similarity:

$$p_{ij} = \frac{\exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{k \neq l} \exp(-\left\|x_k - x_l\right\|^2 / 2\sigma_i^2)} \tag{2.3}$$

Here $p_{i|j}$ is the conditional probability that x_i may pick x_j as a neigbour. This is proportional to the probability density of a Gaussian σ_i centered at x_i.

Perplexity Since we want the number of neighbours of each point to be similar in number and prevent a single point from having a disproportionate influence on the entire system we introduce a hyperparameter named *perplexity*. Perplexity works by ensuring that σ_i is small for points in densely populated areas and large for spare ones and can be thought of as a scale of the number of neighbours considered for any one point in the system. Generally, values between 5 and 50 are considered to give

good results, with larger perplexities taking global features into account, and by consequence smaller ones, local features.

Step 2

Now a probability distribution describing the relationship between points has been formulated, we wish to express this as a low dimensional mapping of our inputs X in terms of our output dimensions Y. Naturally, we would want to make the low dimensional mapping represent a similar (Gaussian) distribution as in Step 1. However, it often causes issues presented by the 'overcrowding problem', Subsection 2.3.3, as the gaussian has a 'short tail', and thus nearby points are likely to be pushed together. A solution to this is the student t-distribution which has a longer tail [5]:

$$q_{i|j} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}} \tag{2.4}$$

Note: *The definition and explination of the Student t-distribution is given in* **??**.

The optimisation of this equation is achieved through the use of *gradient decent*[6] on the Kullback-Leibler divergence **??** between distributions p and q. Here the gradient is used to apply an attractive and repulsive force on the items[7].

2.3.4 Pca Vs T-Sne, A Quick Comparison.

PCA has been around for much longer than t-SNE, and its uses are well established within the scientific community - an example of this would be the use of sensitivity analysis within mechanism reduction [Turanyi and Tomlin, 2015]. It is fast, simple and easy to use and very intuitive. The PCA algorithm works by creating a lower-dimensional embedding which best preserves the overall variance of the dataset. Clusters created from the algorithm are grouped in ways, such that they retain the highest variance of the data.

The main drawback of PCA is that it is a linear projection. If our data happened to be in a 'swiss roll' (spiral) pattern, we would not be able to 'unroll' it. The reason for this is that the PCA algorithm works by viewing the data from different perspectives, much like casting a shadow from various directions. With such an example, there is no one way we can do this that unfurls the spiral.

[5] The distribution employed is a t-distribution with only one degree of freedom and is identical to the Cauchy distribution

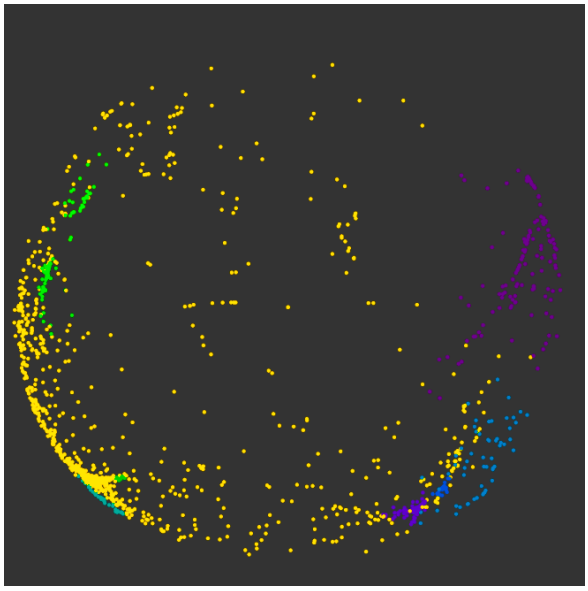
[6] **Gradient Decent** - an optimisation algorithm used to minimise a function by iteratively moving in the direction of the steepest descent. Gradient descent is used to find local minima and is defined by the negative of the gradient of the system. Its primary usage in machine learning is the updating of parameters (coefficients in linear regression and weight in neural networks).

[7] A positive gradient signifies attraction, while a negative one corresponds to repulsion.

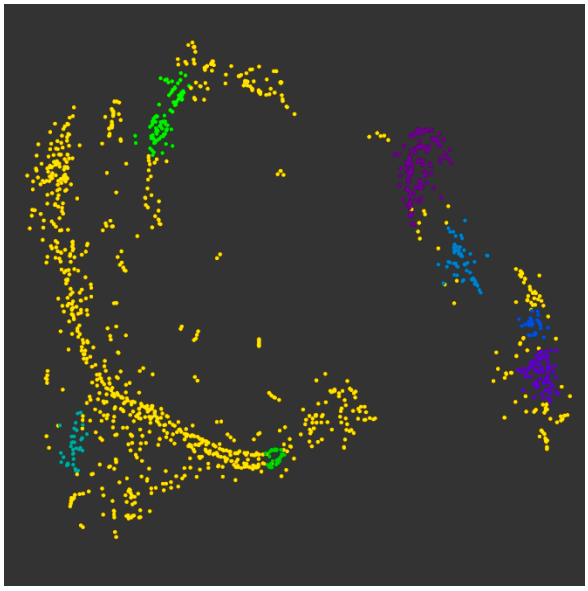
t-SNE, on the other hand, is a relatively new method [Maaten and Hinton, 2008]. Its greatest asset is that linear projections do not limit it. Although more computationally intensive for large datasets, t-SNE produces visibly cleaner results. Unlike in PCA, t-SNE cannot be trained on additional data at a later point; however, the output clusters are more visually distinct (they have less of an overlap). Much like in a force graph, the output from t-SNE is scale-invariant. This means that while the location of clusters in a PCA reduced representation has an attributable quality, those produced by t-SNE will not necessarily contain the same information.

A box model run representative of the chemistry within Beijing was used to compare the differences between PCA and t-SNE. The aim is to classify the diurnal profiles of each species concentration (much like the cosine similarity in **??**). Diurnal profiles were extracted on the third day of a spun up model initialised with initial conditions representative of the chemistry within the Beijing environment (**??**). These were then standardised and converted into temporal vectors for use in the algorithms.

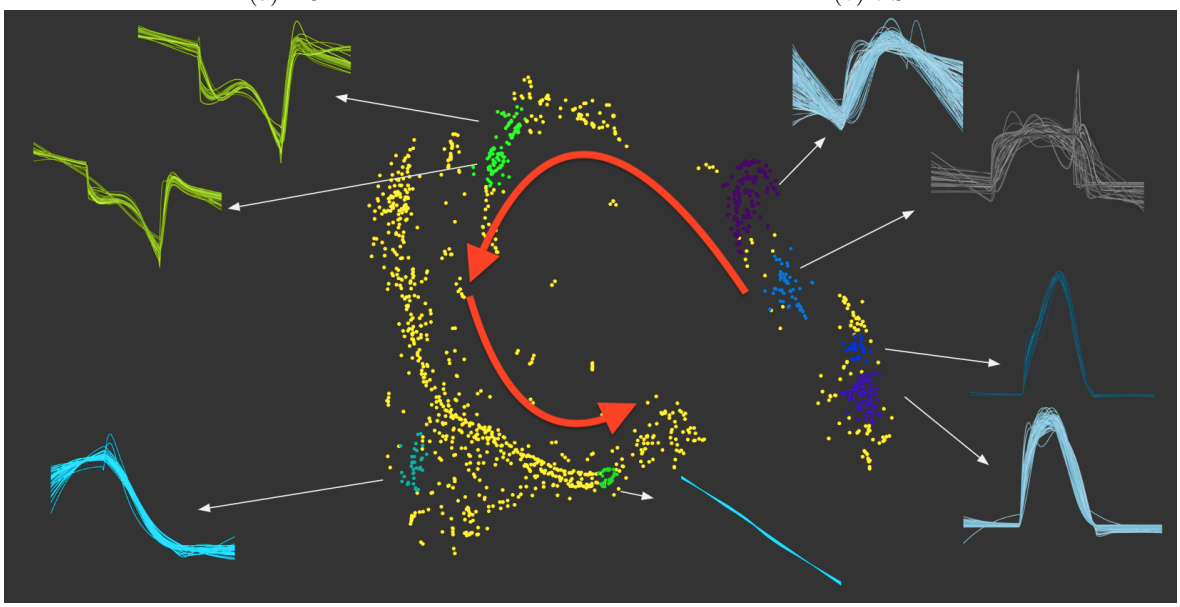
Figure 2.8 shows the output of both dimensionality reduction algorithms on the dataset. Different colours represent the location of clusters of similar diurnal profiles. A higher dispersion between clusters and species overlap is seen within the PCA output, Figure 2.8a. This makes it harder to distinguish species from each other or other groups around them. Since the distance between clusters within t-SNE does not hold the same mathematical meaning as PCA, the algorithm can provide a better distribution of points, creating better-defined clusters, Figure 2.8b. The concentration profile shapes for each coloured group is shown in Figure 2.8c.



(a) PCA



(b) t-SNE



(c) t-SNE with cluster outlines.

Figure 2.8: **Showing the difference between PCA and t-SNE clustering.** These figures show the clustering of a set of standardized concentration profiles (c) across two styles of dimensionality reduction: PCA (a) and t-SNE (b).

2.3.5 The Auto-Encoder (Ae)

Auto-encoders are a subclass of neural networks with primary use in compressing data (dimensionality reduction). Rather than predicting a numerical output, AutoEncoders focus on the construction and deconstruction of data through the use of an encoder and decoder pair. The encoder takes an n-dimensional input and applies a compression, reducing it to the number of dimensions in the bottleneck layer. The reduced dataset is then reconstructed within the decoder. Such a process not only allows for an easy understanding of the error of the reduced data but can also be used in the filtration of

noisy or pixelated data [Leite et al., 2018; Dataman, 2019] and as an input to more complex machine learning models.

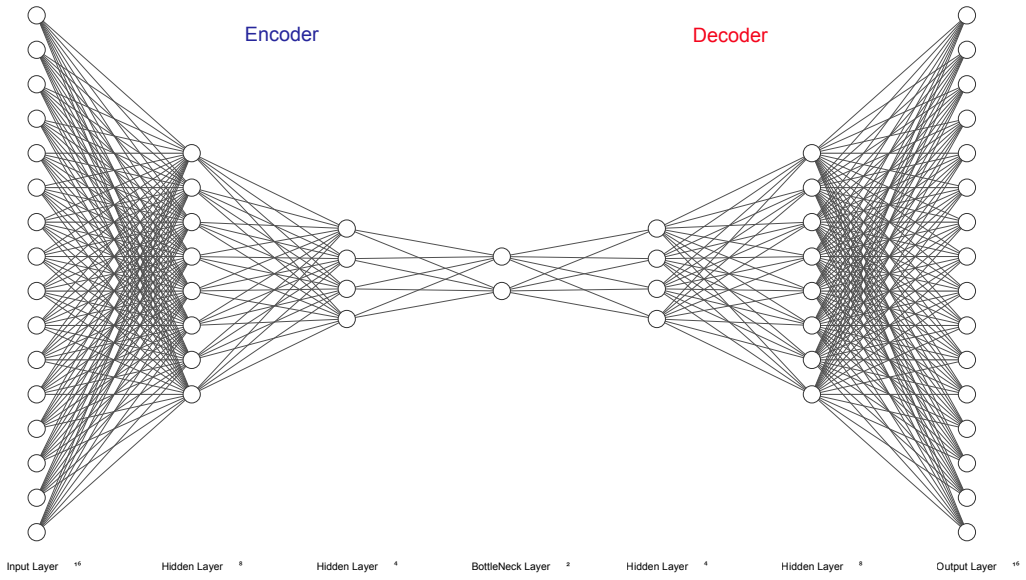


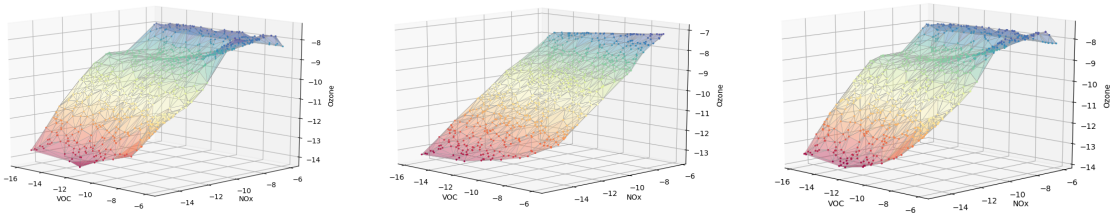
Figure 2.9: An example autoencoder structure which reduces a 16 dimentional input to 2. Draw with the aid of [Krizhevsky et al., 2012]

There are two features of an autoencoder that make it powerful. The first is the ability to sample your latent space using the decoder. The implications of this are that we can establish features that correspond to gaps between our data points - which can have its application if the data used is sparse or incomplete. Next comes the inherent non-linearity of the model. As an autoencoder is just a neural network, the amount of information passed through each link between layers is governed by an activation function. Should this activation function be linear, the reduced dimension will be much akin to a PCA decomposition. Where PCA reduces the dimensions of a dataset by discarding those with a little effect on the variance, an autoencoder opts to combine it- here the entirety of the dataset remains encoded within the links of the AE network. To decide how data flows along the edges of the network, a series of threshold (activation) functions are used for each layer. These are described in **??**.

2.3.5.1 Demonstration Of Non-Linear Activation Functions

To demonstrate the effect of these we take a sample isopleth of Methane and Ozone, reduce it to two dimensions. This is then reconstructed back into three dimensions using the DR algorithms. Figure 2.10 shows the difference between the original dataset (Figure **??**) and that of the PCA (Figure **??**) and AutoEncoder (Figure **??**) reconstructions. Here we see a loss in the non-linearity of the

original data for the PCA reconstruction. However, the use of a non-linear (tanh) activation function within AutoEncoder produces a result much closer to the original. Use of a linear activation function, however, produces a similar result to the PCA algorithm.



(a) Original (b) PCA (c) AE (Tanh)

Figure 2.10: **Comparing the result of the 2D encoding and decoding of an Ozone-NOx-Methane isopleth.** The original data (a) is reduced to two dimensions and then reconstructed back into 3D. This is done with Principal Component Analysis (b) and an AutoEncoder (c). The original isopleth is created using 300 simulations of different intial conditions: NOx (variable), Methane (variable) and Ozone (constant). These were designed using a latin hypercube and converted into a surface plot using Delaunay triangulation.

2.3.6 Node2Vec

Finally, Node2Vec is an embedding algorithm designed to generate vector representations of the nodes in a *undirected* and *unweighted* network. Although it can be used to reduce a complex network into a 2D vector (dimensionality reduction), for this experiment we shall only use it to generate a fingerprint for a species' position within a mechanism network graph - and then apply this as an input to the DR methods above. This method of input creation has been found more computationally efficient, by circumventing the need for expensive composition, in producing better predictions on network-related tasks compared to more classical methods such as PCA [Grover and Leskovec, 2019].

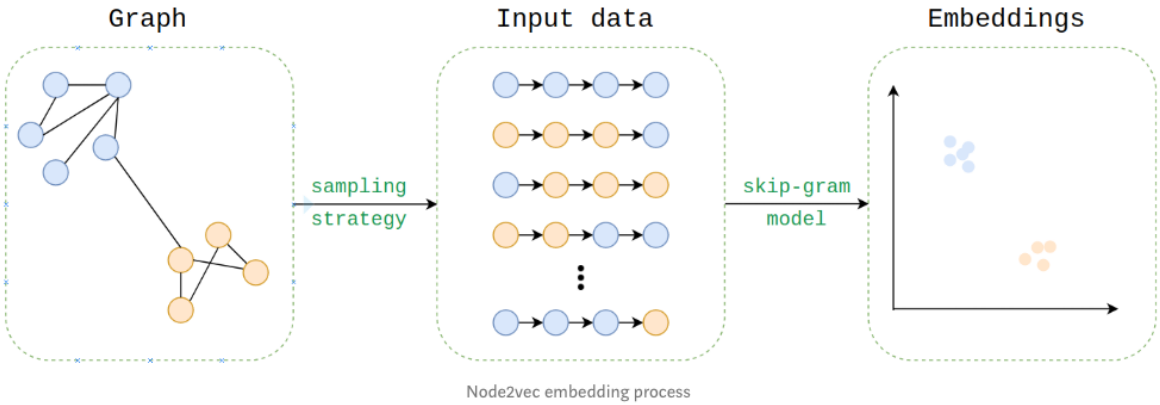


Figure 2.11: The process of converting a graph into a vector using Node2Vec. Source:[Cohen, 2018]

The process of converting the graph structure (Figure 2.11) into a numerical vector node embedding

starts by taking a series of 2^{nd} order random walks. These describe the neighbourhood of a node in the form of a set of random walk paths, much in the same way words are dependant on their neighbours within a sentence: Equation 2.5.

$$ISOPRENE \rightarrow OH \rightarrow TISOPA \rightarrow ISOPBO_2 \rightarrow TISOPA \rightarrow ... \tag{2.5}$$

This methodology allowed for the use of word2vec algorithm, converting the walk into a vector (Subsubsection 2.3.6.2)

2.3.6.1 Sentence Construction By Sampling Of A Network

The probability and path depend both on a set of arguments and a random seed provided to the model. The return and input parameters (p & q) determine how fast we explore the network and our probability to leave the neighbourhood, Figure 2.12. In a system, where the previous path is from t to v, we may calculate the probability of returning to t as $1/p$, going to a mutual node connected between t and v as 1, and viewing a new node as $1/q$. If $q > 1$ we have a high probability to end up at nodes close to t, and with $q < 1$ we are likely to explore other nodes. Additionally if we chose $p > \max q, 1$ we are less likely to return to an already visited node ($p < \min q, 1$ is likely to generate a backwards step). Since we wish to generate a 'local' view, but do not wish to return to t we select $q \geq 1$ and $p > q$ our parameters as $p = 2.0, q = 1.1$. In the case of a weighted graph (something that we are *not* exploring within this chapter) the resultant *alpha* value calculated is further multiplied by the edge weight.

To run the simulation, we use the python2 code provided by the original paper [Grover and Leskovec, 2019] with a set of 50000 random walks, each of length 9. The reasoning behind this is that we have a large graph, with a power-law like structure (where species are often heavily connected, Chapter 1).

NOTE: This process takes over a week to compute (in serial), and then the binary file containing all walks in character form approaches 10 GB, for the complete MCM.

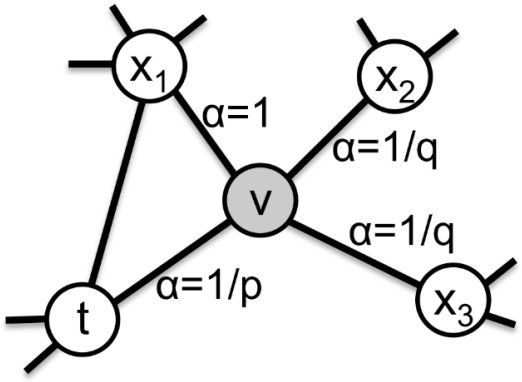


Figure 2.12: Calculation of the random walk path. Source:[Grover and Leskovec, 2019]

2.3.6.2 Word2Vec

Once we have constructed our random path 'sentences' (e.g. Equation 2.5), we can make use of Googles word2vec algorithm [Mikolov et al., 2013]. This is similar to an auto-encoder in many regards; however, the algorithm looks at neighbouring words (or species) in the corpus rather than learning word embeddings using reconstruction. This form of representation has found many uses beyond the realm of natural language processing. Some of these are objects, people, code, tiles,genes and graphs [Lynch, 2011; People2Vec, 2019; Alon et al., 2019; Jean et al., 2018; Du et al., 2019; **?**].

2.3.7 Summary

There exist several methods of reducing a complex dataset into a smaller one. PCA is the simplest method to understand but is constrained to linear decompositions. AutoEncoders can have both a linear and non-linear response, based on the activation functions that they use, and t-SNE applies a non-linear grouping which mimics a complete force-directed graph.

Having defined each method, we next explain how they will be evaluated (Section 2.4), before applying them to the MCM in Equation 2.5.

2.4 Visualisation Of Clustering

In assessing the validity of clustered space, we require a level of exploratory data analysis. To reveal features of interest, we plot the reduced 2D dataset and apply interactivity coupled with a selection of visualisation techniques described below. This section outlines the different visualisation methods which are used.

2.4.1 Viewing The 2D Species Embeddings

Since the different DR algorithms return data on various scales, comparison between the outputs is not straightforward. To overcome this outputs in x and y are normalised (scaled between {0,1}), before being plotted as a scatterplot.

2.4.2 Exposing Overlapping Data

If the nodes within a tight-knit cluster overlap, this can cause obfuscate the results and limit the user's ability to select them. As an initial test, node sizes can be reduced. However, this may often result in points too small to pick. The other solution which was used is to create a force-directed graph where each point is strongly attracted to their initial position. Here we can apply collision detection, while still preserving the overall grouping of nodes within a cluster - a technique that was seen in **??**.

2.4.3 Gooey Effect (Gaussian Blur)

Taking a quote from Reinhardt [1975]: *"The more stuff in it, the busier the work of art, the worse it is. More is less. Less is more. "* and combining it with the work from **??**, we realise that showing each species, when observing overall clusters just add unnecessary clutter to the images. Instead, since we are only interested in the clusters as a unit, a 'gooey effect' filter can be applied. This works by merging nearby points into a single water-like blob using a gaussian blur[8]. Here since each point is allocated a colour, if a colour gradient exists, then there are multiple clusters occupying the same place. The aim of this is to reduce the cognitive load on the end-user by reducing the number of distinct objects that they need to take in.

2.4.4 Four Colours Theorem

When plotted, the number of clusters detected often exceeds the number of categorical colours available. In cartography, it has been noted that the colouring of neighbouring polygons should at most take four colours. This is the origin of the four colours theorem Appel and Haken [1976], of which a greedy implementation is applied.

The aim of this is to show item boundaries (for instance countries, or in our case clusters) while reducing ambiguity (if, say, two neighbours have the same colour). The algorithm I adapted uses the Delaunay tesselation scripts contained within DataDrivenDocuments.js (d3js) Bostock [2012]. This partitions our plane into polygon-regions, each of which includes boundaries at the furthest distance

[8]Here a gaussian blur of standard deviation 3.7 and a colour matrix [1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 37 -5] is used.

from each point (Voronoi cells) Watson [1981]. First, we chose a random cell and assign it a colour. Next, all its neighbours are recursively iterated, giving them the lowest possible colour in a list, which does not match any of their neighbours. Although such a greedy approach does not produce an optimum result, it allows for the colouring of data with ≤ 5 distinct colours, as is shown in Figure 2.13.

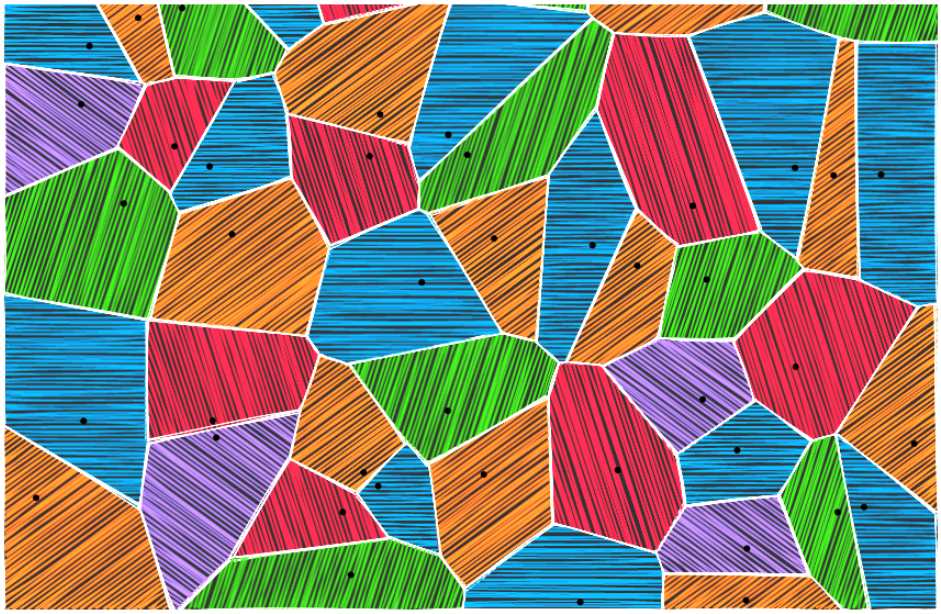


Figure 2.13: **An example 4 colour matching** This uses the first implementation of the algorithm mentioned in Subsection 2.4.4. The greedy approach does not often find the optimum solution, which may result in 5 colours instead. Observable Notebook : Daniel Ellis [2019]

Having defined all the visualisation techniques we move on to explain the clustering algorithms which are used, and how 'goodness of fit' may be measured in the clustering context.

2.5 Cluster Evaluation

The previous section discussed methods of visualising the reduced data for use with interactive exploratory data analysis. In this section we look at the use of vector clustering algorithms[9] (Subsection 2.5.1) to highlight groups in a 2D dataset, as well an automated method of assessing the quality of the clusters selected (Subsubsection 2.5.1.1) and feature extraction (Subsection 2.5.2).

2.5.1 Automated Selection Of Clusters

When it comes to clustering data points in a dataset, there exist a range of methods which may accomplish a task, Figure 2.14. Most often, the k-means [MacQueen, 1967], is used as it is fast and straightforward to understand. However, its linear method of partitioning cannot capture the splits

[9]Vector clustering is the grouping of data based on their proximity or density to other nearby points

between non-linear relationships of real data. The other problem is that an estimate for the number of expected clusters is required - something that is often unknown without prior understanding of the data. When this is the case, often it is easier to select the nodes with interactivity manually.

In contrast, density-based clustering techniques such as GMM ([Pedregosa et al., 2011a]) or DBSCAN ([Ester et al., 1996]) tend to be better at locating non-linear trends in the data. The DBSCAN algorithm asses the distribution of data across a specific location. This allows clusters with a high density of datapoints to be located without the need for a predefined number as an input. Another method: OPTICS (Ordering Points To Identify the Clustering Structure) [Ankerst et al., 1999], shall be used[10]. This is an adaptation of the DBSCAN algorithm which does not require the specification of a minimum distance between points (for the density estimate)- instead, we specify a gradient for the distribution and the minimum number of points for a cluster to be classified.

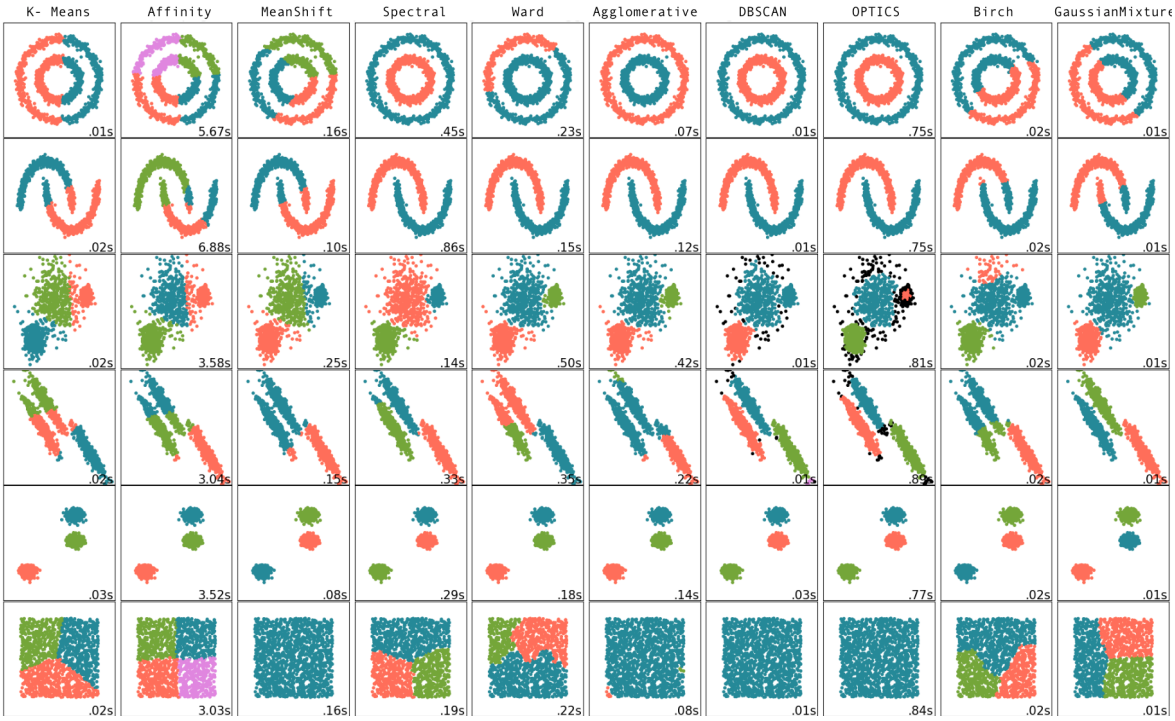


Figure 2.14: **A comparison of different clustering methods on a toy dataset.** The plot shows the performance of several vector clustering algorithms in Scikit-Learn. Cluster algorithms are represented across the horizontal axis and several types of datasets are across the vertical. Clustered groups are coloured. Source: sklearn [2019]

When deciding which algorithms to use, each algorithms' ability to partition non-linear data is considered. The first two rows of Figure 2.14 show data which cannot be partitioned linearly, here spectral, DBSCAN and optics are the only clustering algorithms to identify both correctly. It is for this reason that we shall look at these for the remainder of the chapter.

[10]If using Python 2, the library for this needs to be extracted from the sci-kit-learn library for python3 package and altered to run with the previous version. (See copy in attached code.)

In selecting a value for the results section, several clustering algorithms, with a wide range of input parameters, are run. From these, the simulation with the best silhouette coefficient (Subsubsection 2.5.1.1) is taken.

2.5.1.1 Clustering (Silhouette) Coefficient

The silhouette measure is a tool used for acessing the validity of a set of clusters. Here each cluster is represented as a silhouette, based on the comparison of its tightness and separation. To calculate the silhoette coefficient we look at the intra-cluster a and the mean inter-cluster[11] distance b. The silhouette cluster can then be described using **??**:

$$s(i) = \frac{b(i) - a(i)}{\max a(i), b(i)} \tag{2.6}$$

This gives a value $-1 \leq s(i) \leq 1$. Values near zero suggest overlapping clusters, 1 - dense, well-separated clusters and negative values indicate that a sample may have been incorrectly classified. In using this method, we can get an overview of how well individual objects lie within their assigned cluster.

2.5.2 Feature Extraction

Upon establishing a set of DR datasets, and their groups (the clusters of species they contain), it is important to evaluate what input features they represent. Rather than doing this manually we make use of Random Forests - described below.

2.5.2.1 Random Forrests

Random forests [Breiman, 2001], are a subset of ML algorithms called ensemble learning. This means that they train a large number of decision trees, each on a random subset of the original features. A decision tree is a tree formed from a series of conditionals[12], much like a perceptron network (**??**) with binary activation functions. Random forests introduce a level of additional randomness by selecting only a subset on which to create each decision tree. This may introduce a higher bias, but lowers the overall model variance, which creates a better (more robust) model. Such methods have been applied to replacing the computationally expensive process of chemistry integration of GEOS-Chem (a global 3D model of tropospheric chemistry) [Keller and Evans, 2019] and the prediction of global

[11] Inside and between different clusters.

[12] Questions with a True/False answer

sea-surface iodine based on observations coupled with sea-surface temperature, depth, and salinity [Sherwen et al., 2019].

2.5.2.2 Calculating Importance Using Random Forrests

Since random forests are in essence a collection of decision trees, it is possible to generate a 'decision tree aggregate' to visualise the ensemble structure of the random forest [Ellis and Sherwen, 2019] (Figure 2.15). Alternatively, if all that is required is the relative importance of each feature, the RandomForrestClassifier from Pedregosa et al. [2011b] provides a quick and easy way of understanding which features matter, [Géron, 2017]. This works by aggregating the weighted nodes which use a certain feature using the number of samples and then scales the result to 1. We use this method to access the overall importance of features within each DR output and identify the differences between clusters.

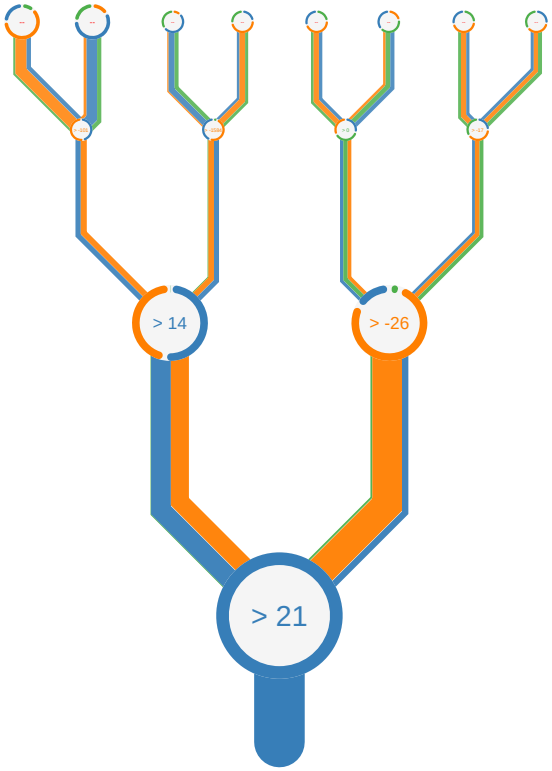


Figure 2.15: **A decision tree aggrogate from a random forrest plotted with the Epiphyte version of the TreeSurgeon program [Ellis and Sherwen, 2019].** The data originates from Sherwen et al. [2019] and the imporance of Tempearature (blue), Depth (orange) and Chlorophyll *a* (green).

NOTE: The only downside is that Random Forrests are in themselves ML techniques which also need to be evaluated. To do this, as they are simply being used as indicators of cluster properties which we are to explore further, we can initiate a collection of 300 random Forrest classifiers, from which we

take the median. A sort of ensemble learning from an ensemble.

2.6 Results

There exist many methods to define the chemical structure of the species within the MCM. In this section, we attempt to evaluate their effectiveness for exhibiting the defining functional groups and characteristics used for constructing the mechanism. First, we explore the distribution of clusters and the ability of different DR algorithms to visually separate various groups of chemistry (Subsection 2.6.1). Next, the functional groups (taken from the MCM development protocol) are explored within each DR algorithm (Subsection 2.6.2). Finally, a selected example for each DR method is taken and explored in further detail (Subsection 2.6.3).

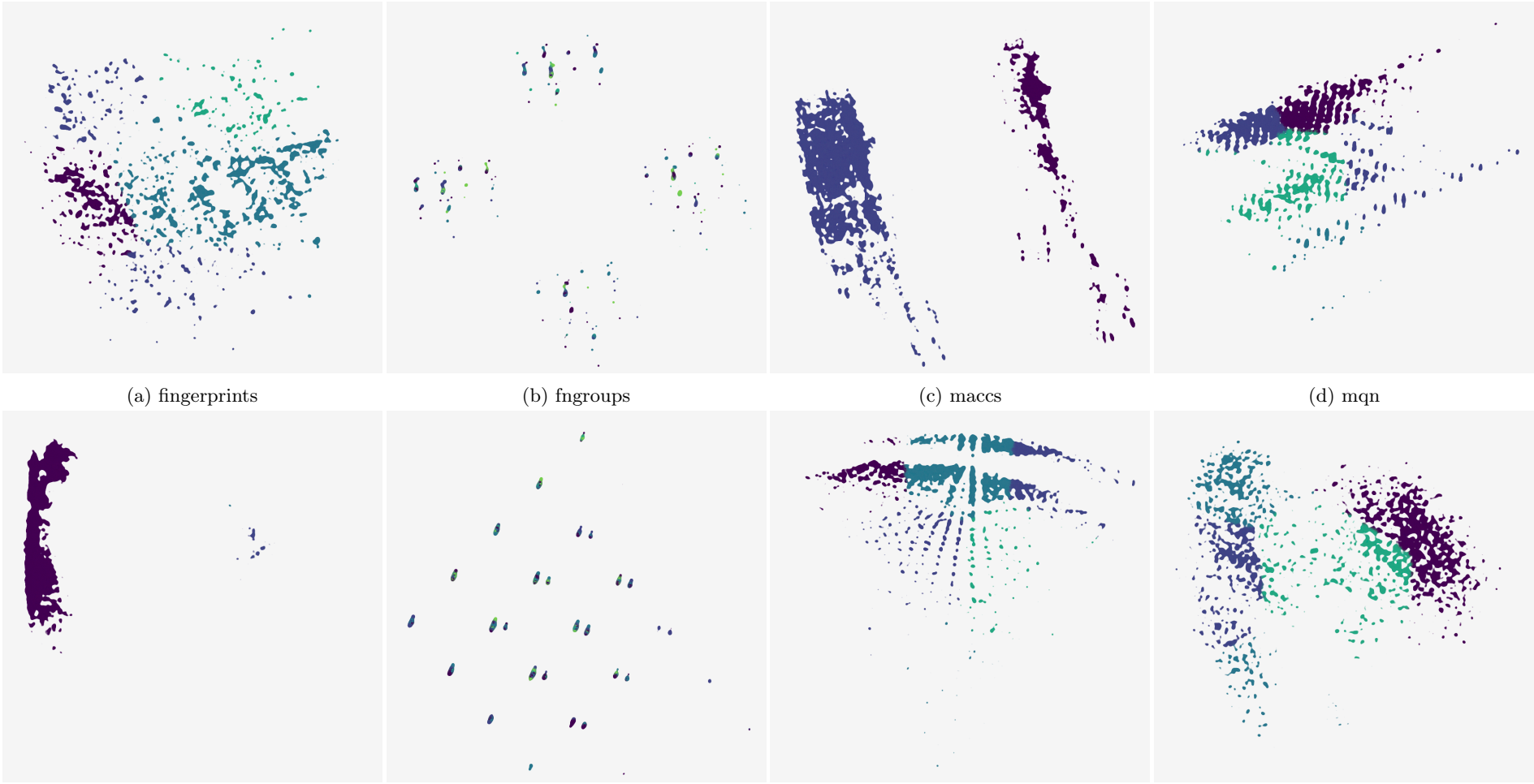
2.6.1 Cluster Distribution

Start with the visual comparison and compare it with the silhouette values.

Principle Component Analysis

DR	input	silhouette	groups
PCA	fngroups	0.9122	141
PCA	protocol	0.8761	149
PCA	node2vec	0.8569	3
PCA	maccs	0.6563	2
PCA	mqn	0.4041	8
PCA	smiles	0.3648	6
PCA	fingerprints	0.3529	6
PCA	spec	0.3364	6

Table 2.2: The inputs to the PCA dimensionality reduction algorithm sorted by the best obtained silhoette coefficient.



(a) fingerprints (b) fngroups (c) maccs (d) mqn

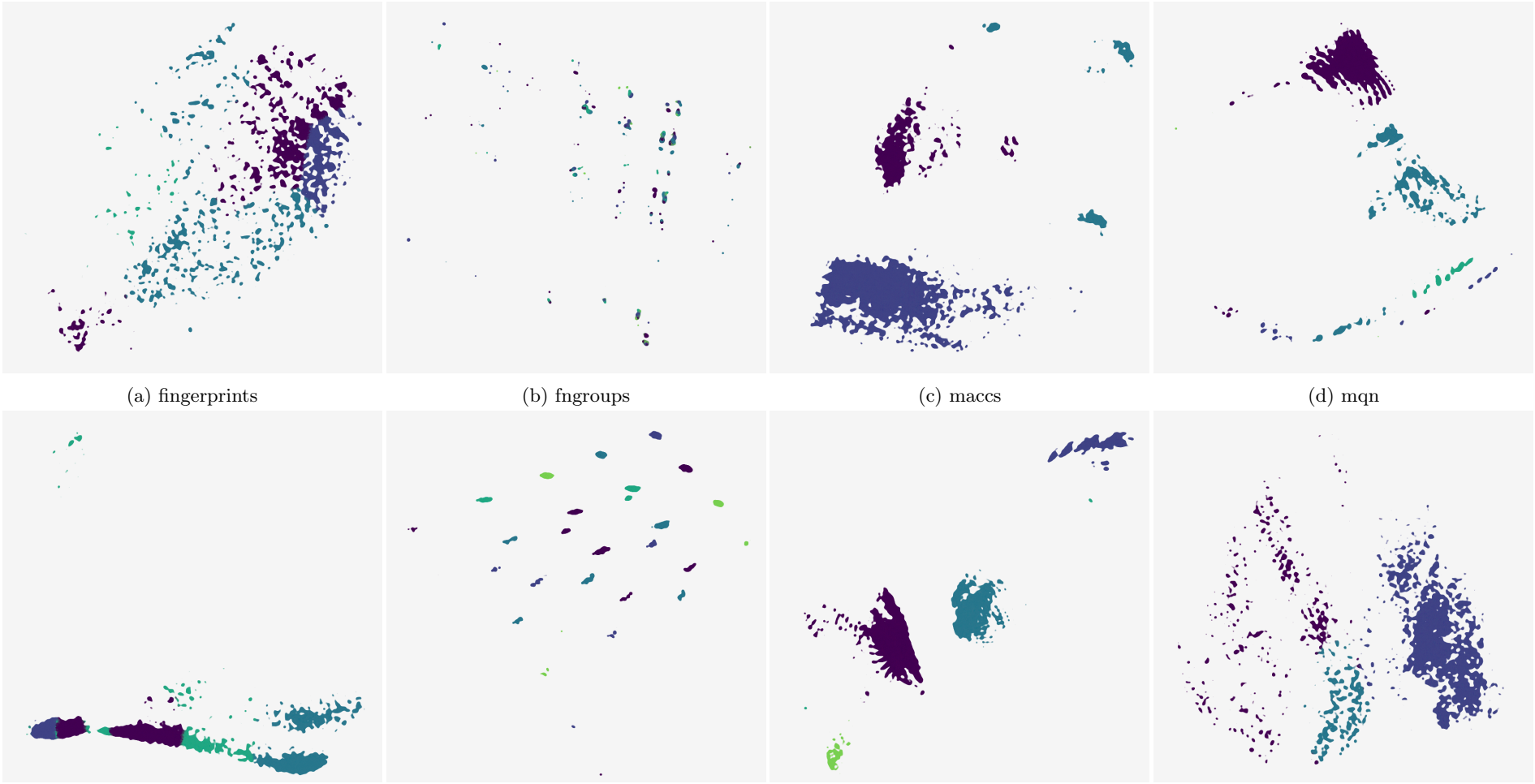
(e) node2vec (f) protocol (g) smiles (h) spec

Figure 2.16: **Comparing clusters for all inputs after a reduction to 2 dimensions using Principle Component analysis.** Each graph has undergone several clustering algorithms under a range of parameters. The result with the best silhouette coefficient has been chosen. Colours follow the greedy four colour theorem and are there only to indicate the contrast between cluster boundaries.

Auto Encoder Encoding

DR	input	silhouette	groups
AE	fngroups	0.9249	140
AE	protocol	0.8992	27
AE	smiles	0.6897	5
AE	mqn	0.6572	12
AE	maccs	0.6241	3
AE	node2vec	0.5476	5
AE	spec	0.4238	3
AE	fingerprints	0.3189	8

Table 2.3: The inputs to the AutoEncoder dimensionality reduction algorithm sorted by the best obtained silhoette coefficient.



(a) fingerprints (b) fngroups (c) maccs (d) mqn

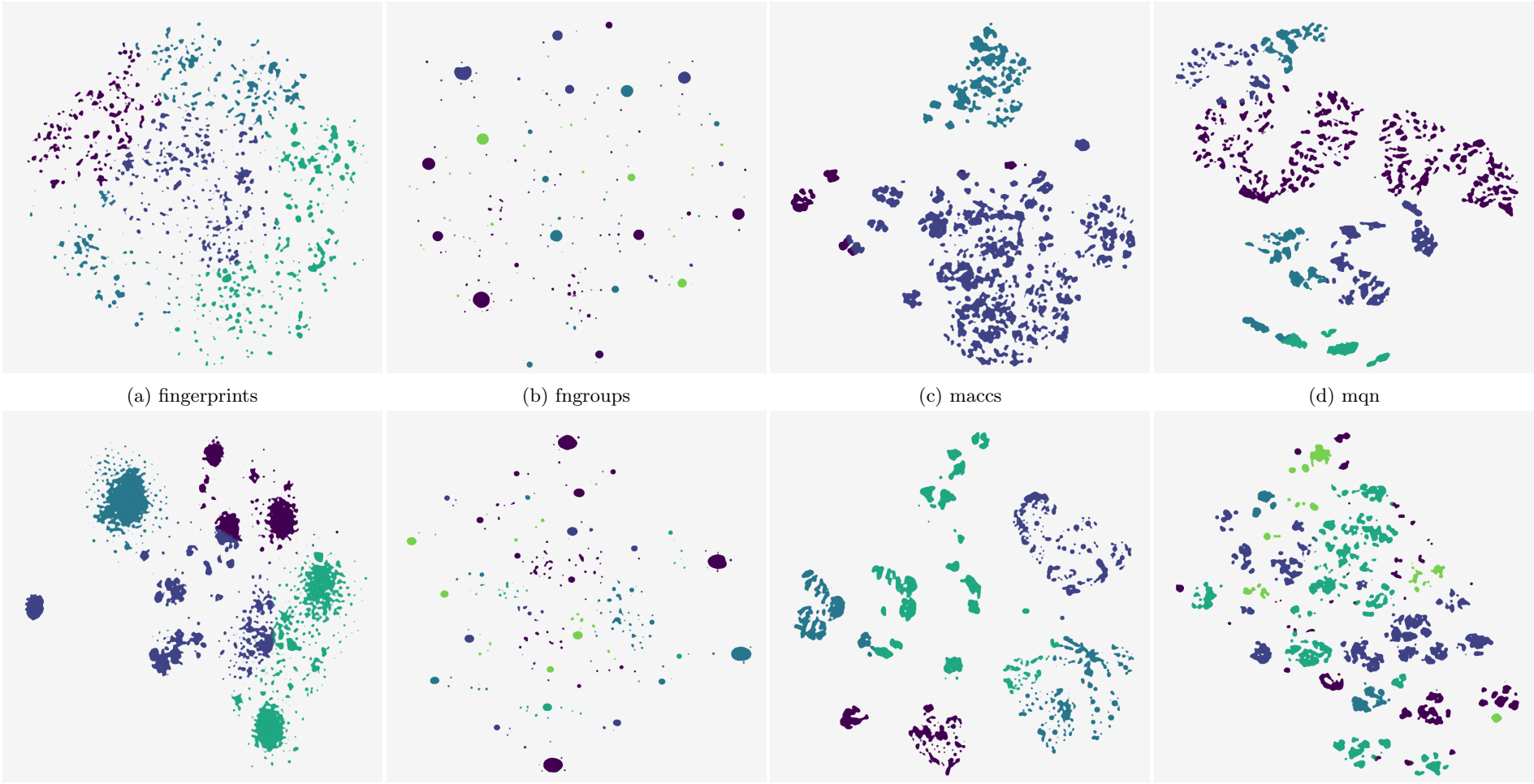
(e) node2vec (f) protocol (g) smiles (h) spec

Figure 2.17: **Comparing clusters for all inputs after a reduction to 2 dimensions using an AutoEncoder.** Each graph has undergone several clustering algorithms under a range of parameters. The result with the best silhouette coefficient has been chosen. Colours follow the greedy four colour theorem and are there only to indicate the contrast between cluster boundaries.

t-Distributed Stochastic Neighbor Embedding

DR	input	silhouette	groups
t-SNE	fngroups	0.7458	106
t-SNE	protocol	0.5688	51
t-SNE	smiles	0.4808	6
t-SNE	node2vec	0.4359	6
t-SNE	maccs	0.4295	3
t-SNE	spec	0.3781	35
t-SNE	mqn	0.3684	8
t-SNE	fingerprints	0.3539	6

Table 2.4: The inputs to the t-SNE dimensionality reduction algorithm sorted by the best obtained silhoette coefficient.

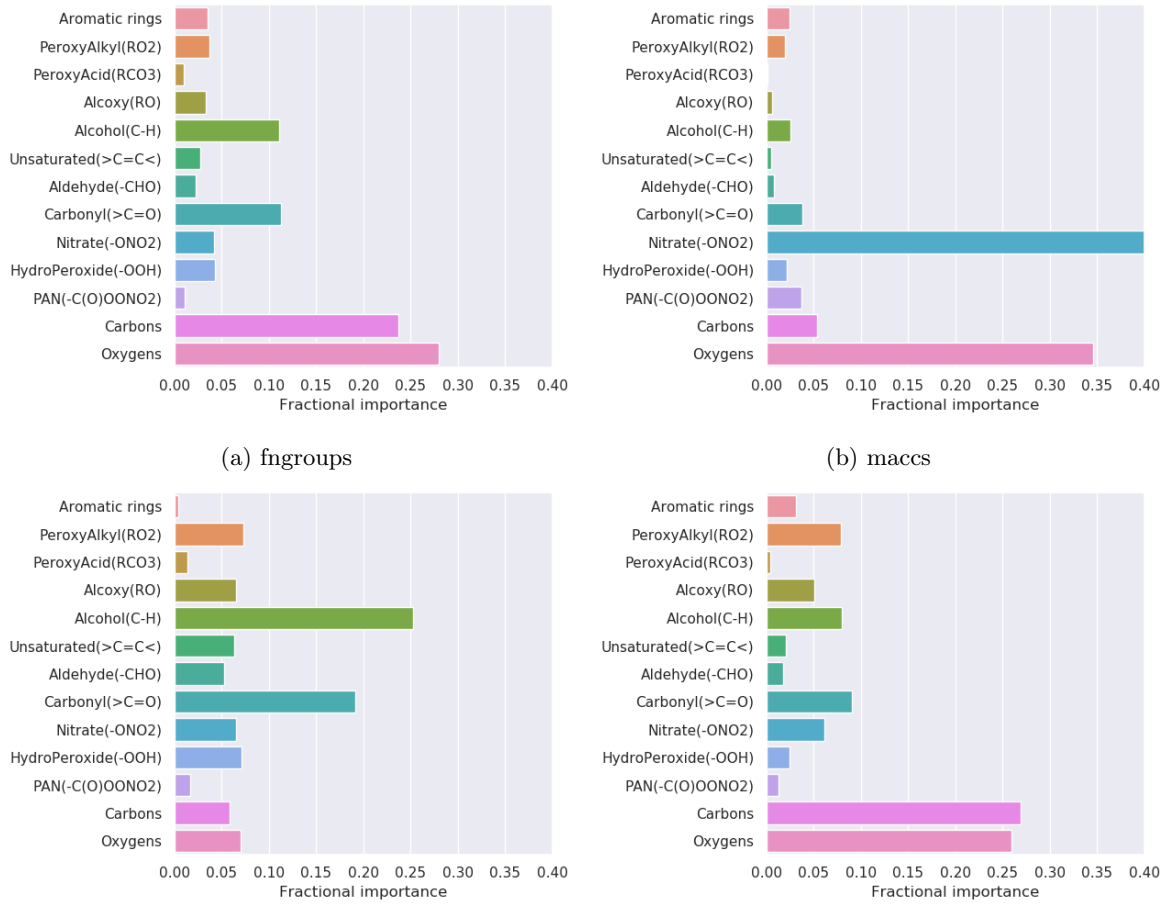


(a) fingerprints (b) fngroups (c) maccs (d) mqn

(e) node2vec (f) protocol (g) smiles (h) spec

Figure 2.18: **Comparing clusters for all inputs after a reduction to 2 dimensions using t-SNE.** Each graph has undergone several clustering algorithms under a range of parameters. The result with the best silhouette coefficient has been chosen. Colours follow the greedy four colour theorem and are there only to indicate the contrast between cluster boundaries.

2.6.2 Feature Selection Comparison



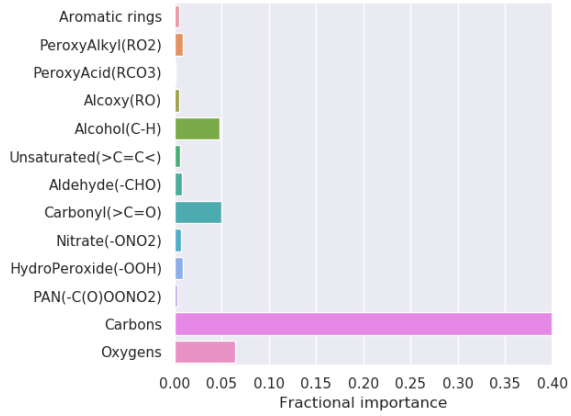
(a) fngroups

(b) maccs

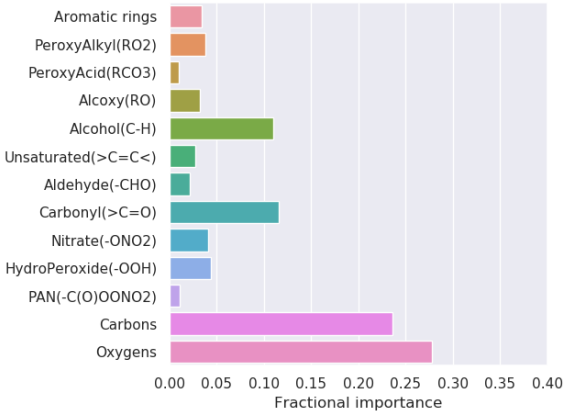
(c) protocol

(d) smiles

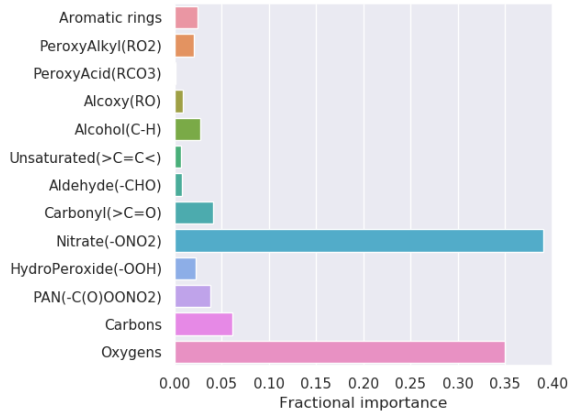
Figure 2.19: **Comparing feature importance for PCA clusters.**



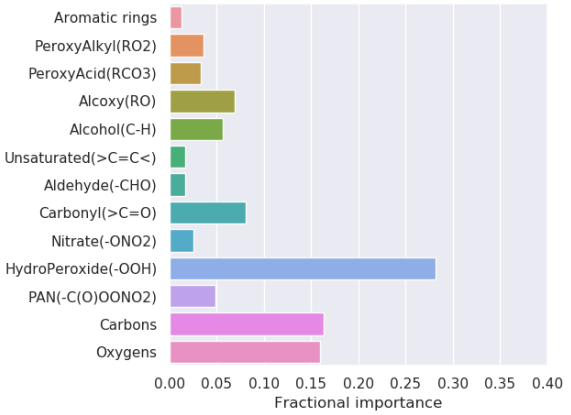
(a) fingerprints



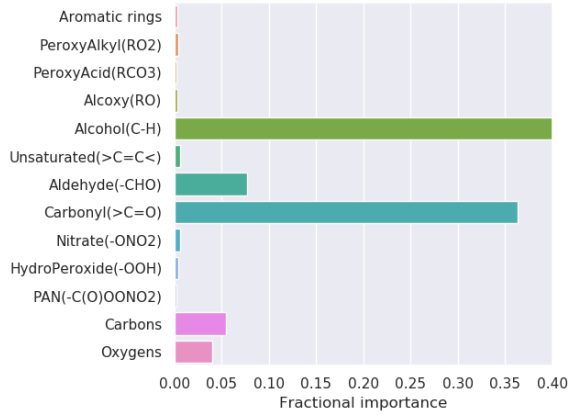
(b) fngroups



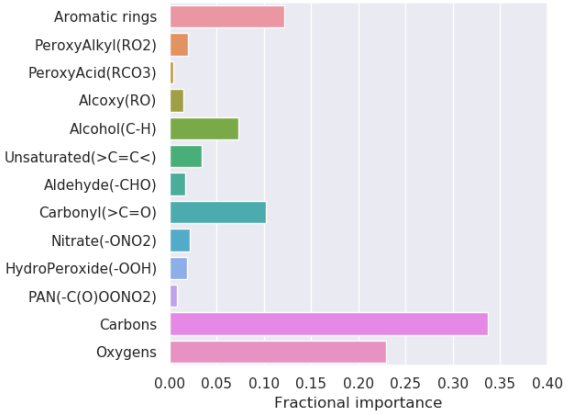
(c) maccs



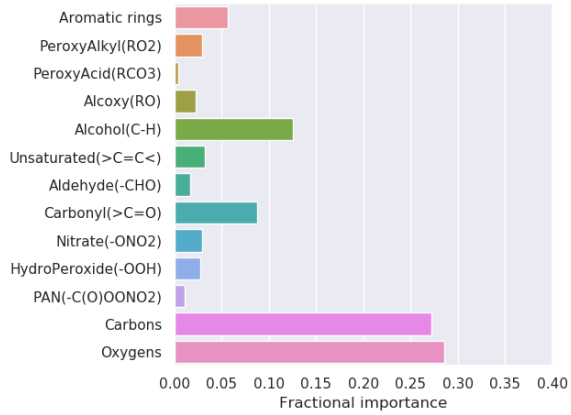
(d) node2vec



(e) protocol

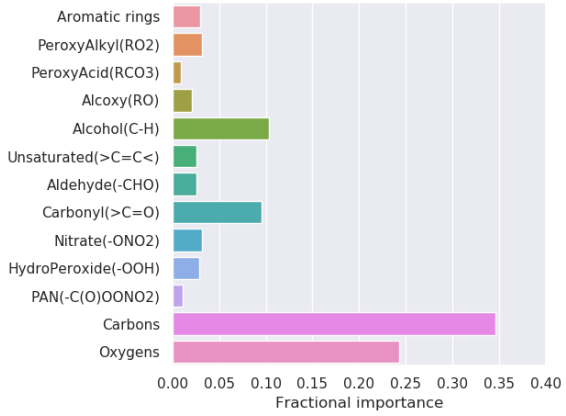


(f) smiles

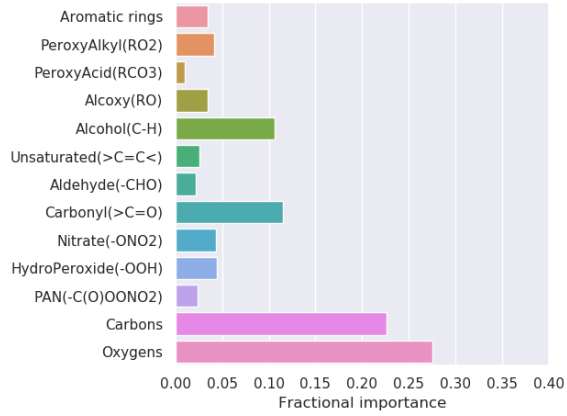


(g) spec

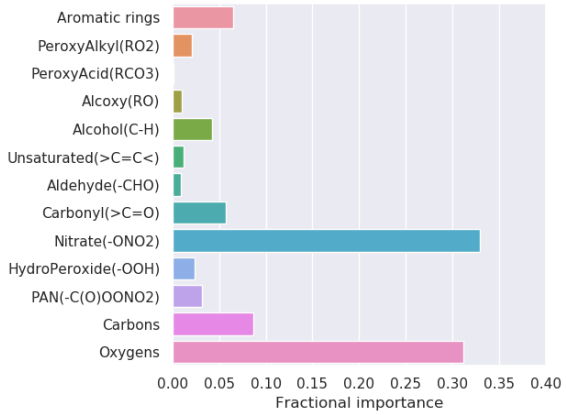
Figure 2.20: **Comparing feature importance for AE clusters.**



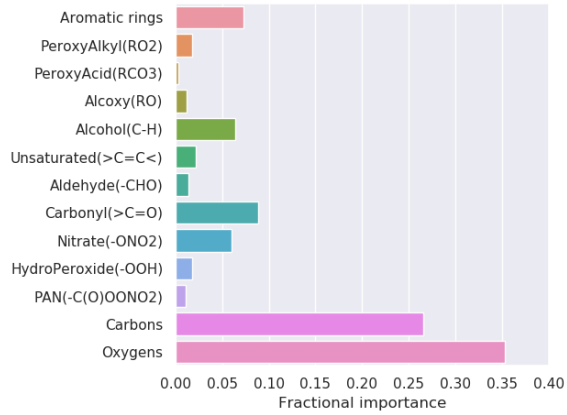
(a) fingerprints



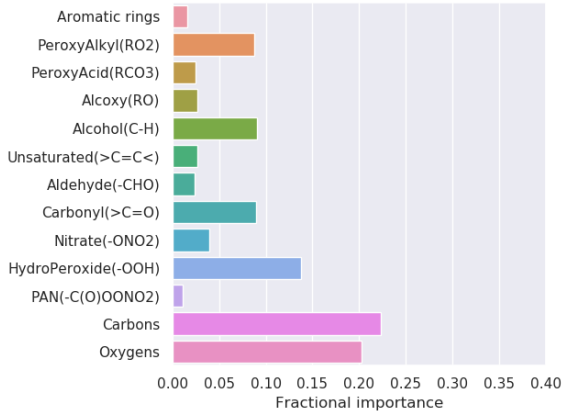
(b) fngroups



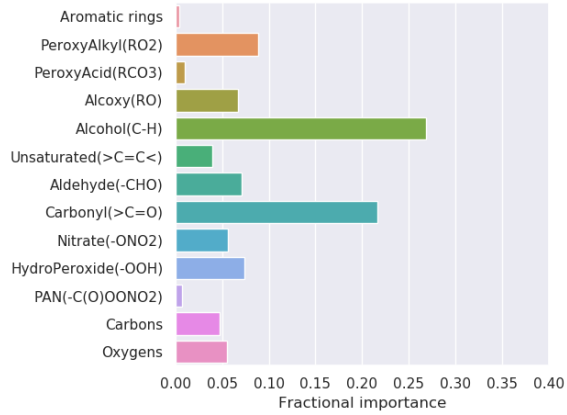
(c) maccs



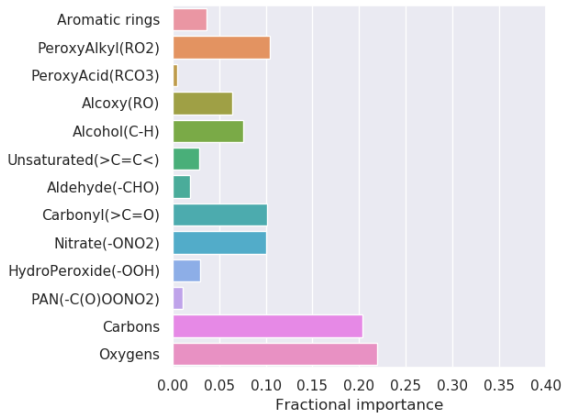
(d) mqn



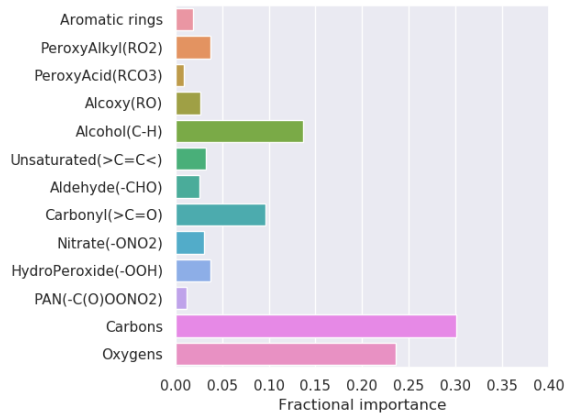
(e) node2vec



(f) protocol



(g) smiles



(h) spec

Figure 2.21: **Comparing feature importance for t-SNE clusters.**

2.6.3 Individual Cluster Comparison

2.7 Conclusions

DR can be used to find patterns in dataset which is best

interaction

There are a range of inputs each showing a few different things.

Depending on what properties we are intereste dwe may select accordingly

Bibliography

Alon, U., Zilberstein, M., Levy, O., and Yahav, E. (2019). Code2Vec: Learning Distributed Representations Of Code. http://dl.acm.org/citation.cfm?doid=3302515.3290353.

Anderson, C. (2008). The End Of Theory: The Data Deluge Makes The Scientific Method Obsolete. *online*. http://www.wired.com/print/science/discoveries/magazine/16-0.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. *SIGMOD Rec.*, 28(2):49–60. https://doi.org/10.1145/304181.304187.

Appel, K. and Haken, W. (1976). Every planar map is four colorable. *Bull. Amer. Math. Soc.*, 82(5):711–712. https://projecteuclid.org:443/euclid.bams/1183538218.

Aumont, B., Szopa, S., and Madronich, S. (2005). Modelling the evolution of organic carbon during its gas-phase tropospheric oxidation: Development of an explicit model based on a self generating approach. *Atmospheric Chemistry and Physics*, 5(9):2497–2517. https://www.atmos-chem-phys.net/5/2497/2005/.

Baillargeon, R. and Carey, S. (2012). Core cognition and beyond: The acquisition of physical and numerical knowledge. *Early childhood development and later outcome*.

Bostock, M. (2012). D3.js - data-driven documents. http://d3js.org/.

Box, G. E. P. (1976). Science And Statistics. *Journal of the American Statistical Association*, 71(356):791–799. https://www.tandfonline.com/doi/abs/10.1080/01621459.1976.10480949.

Breiman, L. (2001). Random Forests. *Machine learning*, 45(1):5–32. https://doi.org/10.1023/A:1010933404324.

Cohen, E. (2018). Node2Vec: Embeddings For Graph Data. https://towardsdatascience.com/node2vec-embeddings-for-graph-data-32a866340fef.

Daniel Ellis (2019). D3-Fourcolour Voronoi. https://observablehq.com/@wolfiex/d3-fourcolour-voronoi.

Dataman (2019). Convolutional Autoencoders For Image Noise Reduction. https://towardsdatascience.com/convolutional-autoencoders-for-image-noise-reduction-32fce9fc1763.

Descartes, R. and Lafleur, L. J. (1960). *Meditations On First Philosophy*. Bobbs-Merrill New York. http://selfpace.uconn.edu/class/percep/DescartesMeditations.pdf.

Du, J., Jia, P., Dai, Y., Tao, C., Zhao, Z., and Zhi, D. (2019). Gene2Vec: Distributed Representation Of Genes Based On Co-Expression. *BMC genomics*, 20(Suppl 1):82. http://dx.doi.org/10.1186/s12864-018-5370-x.

Durant, J. L., Leland, B. A., Henry, D. R., and Nourse, J. G. (2002). Reoptimization Of Mdl Keys For Use In Drug Discovery. *Journal of chemical information and computer sciences*, 42(6):1273–1280. https://www.ncbi.nlm.nih.gov/pubmed/12444722.

Ellis, D. (2019). Chemical Kinetic Interactions Cover Image. https://s100.copyright.com/AppDispatchServlet?startPage=i&publisherName=Wiley&publication=kin&contentID=10.1002%2Fkin.21180&endPage=i&title=Cover+Image%2C+Volume+50%2C+Issue+6.

Ellis, D. and Sherwen, T. (2019). Wolfiex/treesurgeon: Wollemia. https://doi.org/10.5281/zenodo.3346817.

Ester, M., peter Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press.

F.R.S., K. P. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572. https://doi.org/10.1080/14786440109462720.

Géron, A. (2017). *Hands-On Machine Learning With Scikit-Learn And Tensorflow: Concepts, Tools, And Techniques To Build Intelligent Systems*. O'Reilly Media. https://books.google.co.uk/books?id=khpYDgAAQBAJ.

Grover, A. and Leskovec, J. (2019). Node2vec: Scalable feature learning for networks. Accessed: 2019-10-21.

Hamadache, M. and Lee, D. (2017). Principal Component Analysis Based Signal-To-Noise Ratio Improvement For Inchoate Faulty Signals: Application To Ball Bearing Fault Detection. *International journal of control, automation, and systems*, 15(2):506–517. https://doi.org/10.1007/s12555-015-0196-7.

Heller, S., McNaught, A., Stein, S., Tchekhovskoi, D., and Pletnev (2013). Inchi - The Worldwide Chemical Structure Identifier Standard. *Journal of cheminformatics*, 5(1):7. http://dx.doi.org/10.1186/1758-2946-5-7.

Hernandez, W. and Mendez, A. (2018). Application Of Principal Component Analysis To Image Compression. In Göksel, T., editor, *Statistics - Growing Data Sets and Growing Demand for Statistics*. InTech. http://www.intechopen.

com/books/statistics-growing-data-sets-and-growing-demand-for-statistics/application-of-principal-component-analysis-to-image-compression.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441. https://doi.org/10.1037%2Fh0071325.

Jean, N., Wang, S., Samar, A., Azzari, G., Lobell, D., and Ermon, S. (2018). Tile2Vec: Unsupervised Representation Learning For Spatially Distributed Data. http://arxiv.org/abs/1805.02855.

Jolliffe, I. T. and Cadima, J. (2016). Principal Component Analysis: A Review And Recent Developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 374(2065):20150202. http://dx.doi.org/10.1098/rsta.2015.0202.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). Scipy: Open source scientific tools for Python. http://www.scipy.org/.

Keller, C. A. and Evans, M. J. (2019). Application of random forest regression to the calculation of gas-phase chemistry within the geos-chem chemistry model v10. *Geoscientific Model Development*, 12(3):1209–1225. https://www.geosci-model-dev.net/12/1209/2019/.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet Classification With Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Landrum, G., Tosco, P., Kelley, B., sriniker, gedeck, NadineSchneider, Vianello, R., Dalke, A., Cole, B., AlexanderSavelyev, Turk, S., Ric, Swain, M., Vaucher, A., N, D., Wójcikowski, M., Pahl, A., JP, strets123, JLVarjo, O'Boyle, N., Berenger, F., Fuller, P., Jensen, J. H., Sforna, G., DoliathGavid, Cosgrove, D., Nowotka, M., Leswing, K., and van Santen, J. (2019). Rdkit 2019-03-2 (q1 2019) release. https://doi.org/10.5281/zenodo.2864247.

Leite, N. M. N., Pereira, E. T., Gurjão, E. C., and Veloso, L. R. (2018). Deep convolutional autoencoder for eeg noise filtering. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2605–2612.

Lynch, H. (2011). *Infant Places, Spaces And Objects: Exploring The Physical In Learning Environments For Infants Under Two*. PhD thesis. http://dx.doi.org/10.21427/D73W37.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing Data Using T-Sne. *Journal of machine learning research: JMLR*, 9(Nov):2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press. https://projecteuclid.org/euclid.bsmsp/1200512992.

(MDL), M. I. S. (1984). Maccs-ii.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation Of Word Representations In Vector Space. http://arxiv.org/abs/1301.3781.

Morozov, A. (2016). Modelling biological evolution: Linking mathematical theories with empirical realities. *Journal of Theoretical Biology*, 405:1 – 4. http://www.sciencedirect.com/science/article/pii/S0022519316301849.

Nguyen, K. T., Blum, L. C., van Deursen, R., and Reymond, J.-L. (2009). Classification Of Organic Molecules By Molecular Quantum Numbers. *ChemMedChem*, 4(11):1803–1805. http://dx.doi.org/10.1002/cmdc.200900317.

Noble, C. E. (1957). Human Trial-And-Error Learning. *Psychological reports*, 3(2):377–398. https://doi.org/10.2466/pr0.1957.3.h.377.

Oliphant, T. (2006). Guide to numpy.

Oliveira, B., Pereira, F., de AraÃ°jo, R., and Ramos, M. (2006). The hydrogen bond strength: New proposals to evaluate the intermolecular interaction using dft calculations and the aim theory. *Chemical Physics Letters*, 427(1):181 – 184. http://www.sciencedirect.com/science/article/pii/S000926140600861X.

Parsons, J., Holmes, J. B., Rojas, J. M., Tsai, J., and Strauss, C. E. M. (2005). Practical Conversion From Torsion Space To Cartesian Space For In Silico Protein Synthesis. *Journal of computational chemistry*, 26(10):1063–1068. http://dx.doi.org/10.1002/jcc.20237.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011a). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830. http://dl.acm.org/citation.cfm?id=1953048.2078195.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011b). Scikit-Learn: Machine Learning In Python . *Journal of Machine Learning Research*, 12:2825–2830.

People2Vec (2019). People2Vec. http://people2vec.org/.

Powell, V. (2020). Principal Component Analysis Explained Visually. http://setosa.io/ev/principal-component-analysis/.

Probst, D. and Reymond, J.-L. (2018). Smilesdrawer: Parsing And Drawing Smiles-Encoded Molecular Structures Using Client-Side Javascript. *Journal of chemical information and modeling*, 58(1):1–7. http://dx.doi.org/10.1021/acs.jcim.7b00425.

rdkit (2019). Rdkit. https://github.com/rdkit/rdkit/blob/24f1737839c9302489cadc473d8d9196ad9187b4/rdkit/Chem/MACCSkeys.py.

Reinhardt, A. (1975). *Art-As-Art: The Selected Writings Of Ad Reinhardt*. Documents of 20th-century art. Viking Press. https://books.google.co.uk/books?id=zyK4AAAAIAAJ.

Roberts, R. (1989). *Serendipity: Accidental Discoveries In Science*. Wiley Science Editions. Wiley. https://books.google.co.uk/books?id=hf57X0s4aPwC.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326. https://science.sciencemag.org/content/290/5500/2323.

Sherwen, T., Chance, R. J., Tinel, L., Ellis, D., Evans, M. J., and Carpenter, L. J. (2019). A machine-learning-based global sea-surface iodide distribution. *Earth System Science Data*, 11(3):1239–1262. https://www.earth-syst-sci-data.net/11/1239/2019/.

sklearn (2019). Comparing Different Clustering Algorithms On Toy Datasets — Scikit-Learn 0.21.3 Documentation. https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html.

Spahn, V., Del Vecchio, G., Labuz, D., Rodriguez-Gaztelumendi, A., Massaly, N., Temp, J., Durmaz, V., Sabri, P., Reidelbach, M., Machelska, H., Weber, M., and Stein, C. (2017). A Nontoxic Pain Killer Designed By Modeling Of Pathological Receptor Conformations. *Science*, 355(6328):966–969. http://dx.doi.org/10.1126/science.aai8636.

T. Leube, B., Inglis, K., J. Carrington, E., and Sharp, P. (2018). Lithium transport in li 4.4 m 0.4 m Â 0.6 s 4 (m = al 3+ , ga 3+ and m Â= ge 4+ , sn 4+): Combined crystallographic, conductivity, solid state nmr and computational studies. *Chemistry of Materials*, 30.

Turanyi, T. and Tomlin, A. (2015). *Analysis Of Kinetic Reaction Mechanisms*. Springer. http://eprints.whiterose.ac.uk/84294/.

Wang, S.-G. and Schwarz, W. H. E. (2009). Icon Of Chemistry: The Periodic System Of Chemical Elements In The New Century. *Angewandte Chemie*, 48(19):3404–3415. http://dx.doi.org/10.1002/anie.200800827.

Watson, D. F. (1981). Computing The N-Dimensional Delaunay Tessellation With Application To Voronoi Polytopes*. *The Computer Journal*, 24(2):167–172. https://doi.org/10.1093/comjnl/24.2.167.

Weininger, D. (1988). Smiles, A Chemical Language And Information System. 1. Introduction To Methodology And Encoding Rules. *Journal of chemical information and computer sciences*, 28(1):31–36. https://pubs.acs.org/doi/abs/10.1021/ci00057a005.

Yu-ChenLo (2018). Machine learning in chemoinformatics and drug discovery. *Drug Discovery Today*, 23(8):1538 – 1546. http://www.sciencedirect.com/science/article/pii/S1359644617304695.