Master's Dissertation in Design and Multimedia

# Visualization for Genetic Algorithms

António Malta Lopes da Cruz
*antonioc@student.dei.uc.pt*

Supervisors:
Penousal Machado
Pedro Miguel Cruz

Department of Informatics Engineering
Faculty of Sciences and Technology of the University of Coimbra

# Abstract

Information visualization is still evolving alongside the incredible technological advances of the past decades as we can now use computers to process exceptionally large datasets easily and represent them through animated and interactive visualizations. The development of this field has been largely dependent on its work with other fields by finding new ways to visualize data and finding meaningful patterns of information. One of these fields is artificial intelligence, which often turns to nature for inspiration in problem-solving and devises tools such as the genetic algorithm, a search heuristic which simulates natural selection in order to find and optimize solutions to particular problems.

In this dissertation we covered some of the most important developments from the fields of information visualization and genetic algorithms, and detail the process of the creation of a new visualization tool. This takes the form of a functional prototype which can process the data acquired from a genetic algorithm and, using visualization techniques which had not been applied to this particular field before, is able to effectively represent meaningful patterns in the data which lead to significant conclusions.

# Keywords

Adaptive visualization, emergence in visualization, information visualization, natural selection, sexual selection.

# Acknowledgements

Dedicated to my mother, father and brother who I love very much and who have always been there for me, to all of my family who have always been kind and supportive, and to my supervisors who put up with me and without whom this would not have been possible.

# Contents

# Chapter 1
# Introduction

Information visualizations were created and have continued to evolve due to the necessity to communicate through images since this allows for visual representations of various amounts of information, as well as of the connections between these various elements, and generally convey these in a simple, comprehensible visualization.

*"Graphic representation constitutes one of the basic sign-systems conceived by the human mind for the purpose of storing, understanding, and communicating essential information. As a "language" for the eye, graphics benefits from the ubiquitous properties of visual perception."*

– Jacques Bertin, 1967, Semiology of Graphics [1], p.2

Proper visualizations can be very effective in both storing data and making it simple to search through for meaningful information, which can be interpreted from emerging patterns or even specific values, elements or even relationships that become easier to understand or follow when properly represented. Information design is not an exact science, but certain theoretical principles have been established over the years by people who have excelled in the field, such as Jacques Bertin [1] and Edward Tufte [2], who have defined some of the most basic and more complex rules for properly encoding and organizing information through the study and understanding of cognitive and visual perception. These principles have also been evolving over the years alongside the visualizations themselves, as both of these have dramatically changed through the advent of new technologies, and continue to change and evolve even now.

The fast processing of computer has allowed us to extend beyond static images, as their ability to generate multiple images in spans of time shorter than a second has allowed us to create interactive and dynamic displays. These computational processes have become an almost essential tool in analyzing the incredibly large amounts of data from the digital databases which are constantly being created and expanding. It is in our interest, as both designers and researchers, to make use of these tools in order to provide proper visualizations that can induce useful insights into this collected information, as otherwise any conclusions would have to be drawn from the purely textual and numerical data which would make the process a lot slower an ineffective. Dynamic visualizations have shown to be very effective and have a lot of potential as they introduce elements animation and interaction. Animations on their own add a whole new dimension to the visualization (usually representing time), which otherwise would have

to be represented using one of the planar coordinates or multiple iterations of the visualization. Interaction allows for the user to directly change the visualization on the spot, though this is restricted to the options that were programmed. These functions can be used to explore the data in various ways, such as changing the level of detail, which includes zooming in and out or even controlling the quantity of information on the screen, changing the visual aspects, such as the colors and shapes used or even the structure of the visualization itself, and even navigation functions which let the user view sets of information which might be too large to be displayed at once. However, there are still constantly new approaches being taken to tackle dynamic visualizations, both on how to represent dynamic data, and on the interactive tools themselves.

Our objective stands on understanding and using these new tools to build new ones which can be used to help visualize and understand large amounts of data, although this is a very broad description. Representing data properly relies heavily on both the type of data itself and its background, since the visualization must be built within this context or it could easily be misinterpreted, and because of this our intentions are to focus on a particular area rather than building a tool which could represent a much wider range of datasets. We focused on data collected from genetic algorithms, which are, in essence, a tool from the field of artificial intelligence used for finding and optimizing solutions to certain problems. Like many other modern tools, genetic algorithms were inspired in nature, more particularly in the process of natural selection where the fittest living beings are more likely to survive, breed and pass on their genes.

These algorithms use an iterative process which involves evaluating and recombining solutions in order to possibly obtain better ones and there are a number of variable parameters that influence the outcome and these must be carefully adjusted depending on the problem itself. This process can be run for hundreds or even over thousands of cycles in order to reach one of the best possible or most optimal solutions, and these final outputs are usually the most important parts of the data, however our objective isn't to display or analyze the best results but rather the process which lead to them. As it follows, we intend on building and developing an interactive visualization tool (a computer application) which can receive data obtained from genetic algorithms which details their solution searching process, and then properly represent it on screen in a way which is easy to understand and to interpret. Visualizing all the cycles and alterations that take place between each one is a task best suited for an interactive application because it should allow users to efficiently search through the data intuitively and provide simple functions which gives them more control over the visualization such as data filters.

Despite our focus on constructing a tool to visualize datasets of a specific nature, this project encompasses a variety of fields which all have contributed to its creation and development, which are described in the state of the art. The state of the art includes: a brief history of information visualization describing some of the more important and varied works and

achievements in the field; an analysis on proper and effective representation of information; a basic understanding of the visualization structures and representation techniques which we will be approaching; the history, purpose and behavior description of a genetic algorithm, which provides the context necessary to understand both the nature and significance of the data and resulting visualizations. The chapters that follow the state of the art entail: a detailing of the project's objectives and the work plan followed throughout the development, along with the chosen methodologies; a description of all the preparatory work accomplished prior to the development of the application and includes a project created for a contest held by the Massachusetts Institute of Technology; the development process of this dissertation's project which describe all of the implemented functions, the resulting visualizations, and an overview of the results, along with our conclusions.

# Chapter 2
# State of the art

## Brief history of information visualization

Information visualization has been developed through countless contributions, both theoretical and practical. In this chapter will cover some of the most important and significant creations in order to understand the evolution alongside the technological progress of the past centuries and its reach into other fields of investigation.

The idea of making maps and tabular presentations can be dated as far as the 200 BC [3], when it was used by Egypt for astrology and navigation. This anonymous tenth century graph (Fig. 2.1) is one of the earliest known graphical depictions of quantitative information with multiple variables, by representing the position of the sun, moon and planets throughout the year. However, it wasn't until the seventeenth century that the representation of quantitative information on a two-dimensional plane would really start to develop. During the seventeenth century, some of the most prominent problems were the physical measurement of time, space and distance, mostly concerning areas such a astronomy, map creation, navigation and territorial expansion. It was also a century that witnessed the development a lot of theoretical work related to geometry and coordinate systems, error measurement and estimations, probability, demography, and statistics as a whole. One of such examples was Christopher Scheiner's *Sunspots* (Fig. 2.2), which depicted changes in sunspots over time while using an idea which Edward Tufte much later called "small multiples" [2], where various iterations of the visualization are shown in order to represent changes over time. The century gave way to the beginnings of visual thinking, in which people looked to make sense of real and interesting data by representing it visually in order to be able to interpret significant insights.

New graphic forms started appearing with the eighteenth century, as abstract graphs and graphs of functions became more prevalent and new data representations were invented. The interest also turned to the collection of data from areas such as politics and the economy which also shifted the handling how this information should be communicated, leading to a search for novel visual forms of representation. Cartography began exploring the thematic mapping of geological, economic and medical data.

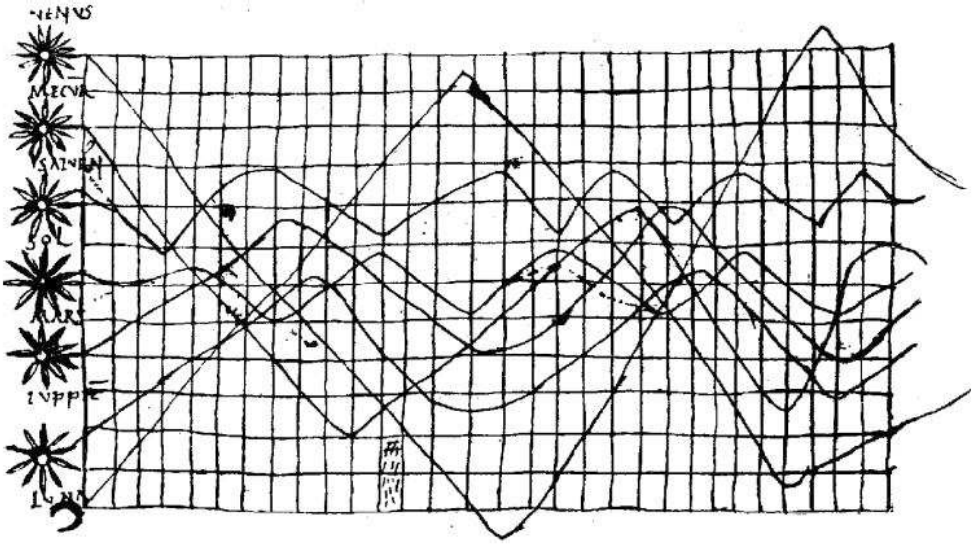



**Fig. 2.1** *Heavenly Bodies* (Unknown author)
Graph from the tenth century showing the changes in the positions of the sun, moon and planets over the course of a year.

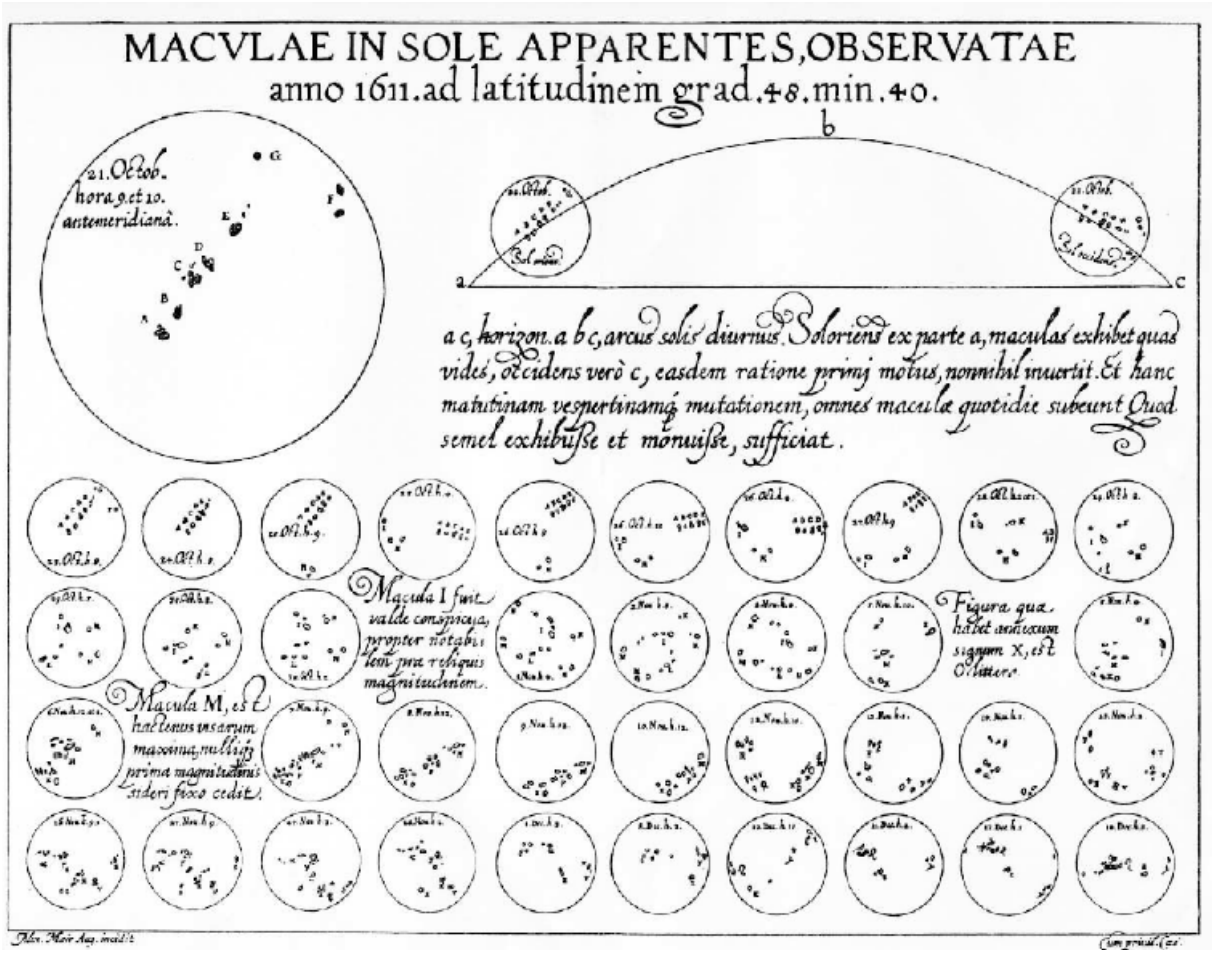



**Fig. 2.2** *Sunspots* (Christopher Scheiner, 1626)
A graph showing the changes in sunspots over time. It's one of the earliest known implementations of the concept of "small multiples".

Johann Lambert (1728–1777), a Swiss-German scientist and mathematician, along with William Playfair (1759–1823), a Scottish political economist, are considered to be two of the great inventors of modern graphical design. Lambert's background in mathematics allowed for a clearer approach to abstract problems of graphical design, which was apparent in his work such as some of his experiments with heat shown in *Pyrometrie* (Fig. 2.3), published in 1779. Meanwhile, Playfair became the creator of most of the graphical forms used today, as he popularized some of the most fundamental forms of statistical graphs (such as the bar graph and the pie chart). Playfair considered graphics to be preferable to tables as they allow for better comparisons between the data values and different information.

Abstracting the map coordinates and moving away from geography was a big step taken by great creators of the eighteenth century, such as Playfair and Lambert, but even then they were using analogies to the physical world as a base. With the evolution of their work, graphical design moved away from these analogies and became no longer dependent on them.
Data did not have to be tied to geographic or time coordinates and could be placed in relationship to other variables. Playfair's Commercial and Political Atlas released in 1786 was the first publication to contain statistical charts but no maps, and it included sets of time-series charts depicting different kinds of information, such relations between exports and imports between countries (Fig. 2.4), as well as one the first simulaneous use of a bar chart and a line chart with economical data which simultaneously represented three different time-series (Fig. 2.5).

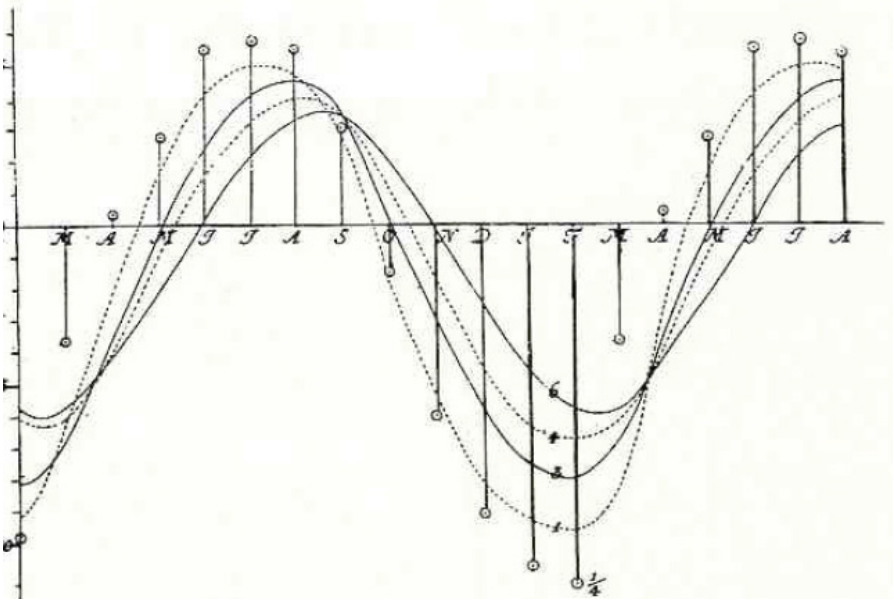


**Fig. 2.3** *Pyrometrie* (Johann Lambert, 1779)
Chart is taken from the series Pyrometrie which depicts the periodic variation of the soil temperature in relation to depth.

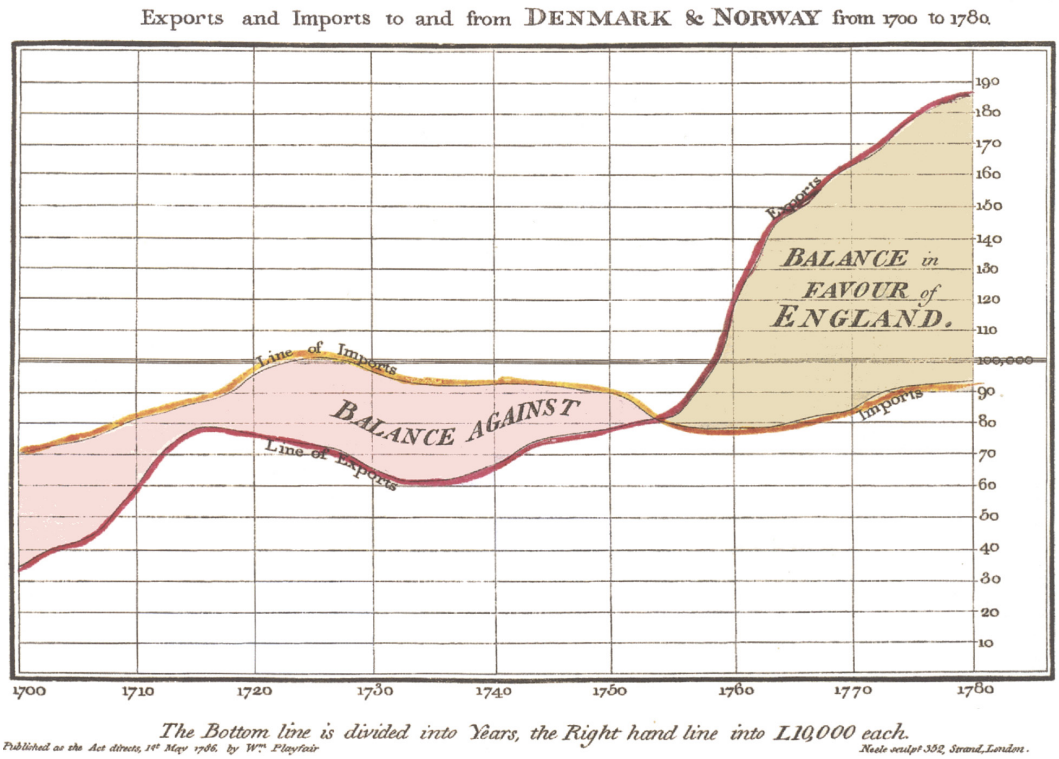



**Fig. 2.4** *Exports and Imports to and from Denmark & Norway from 1700 to 1780* (William Playfair, 1786)
A time-series chart that uses the area between the lines representing exports and imports to quantify how favorable or not the balance is towards England.

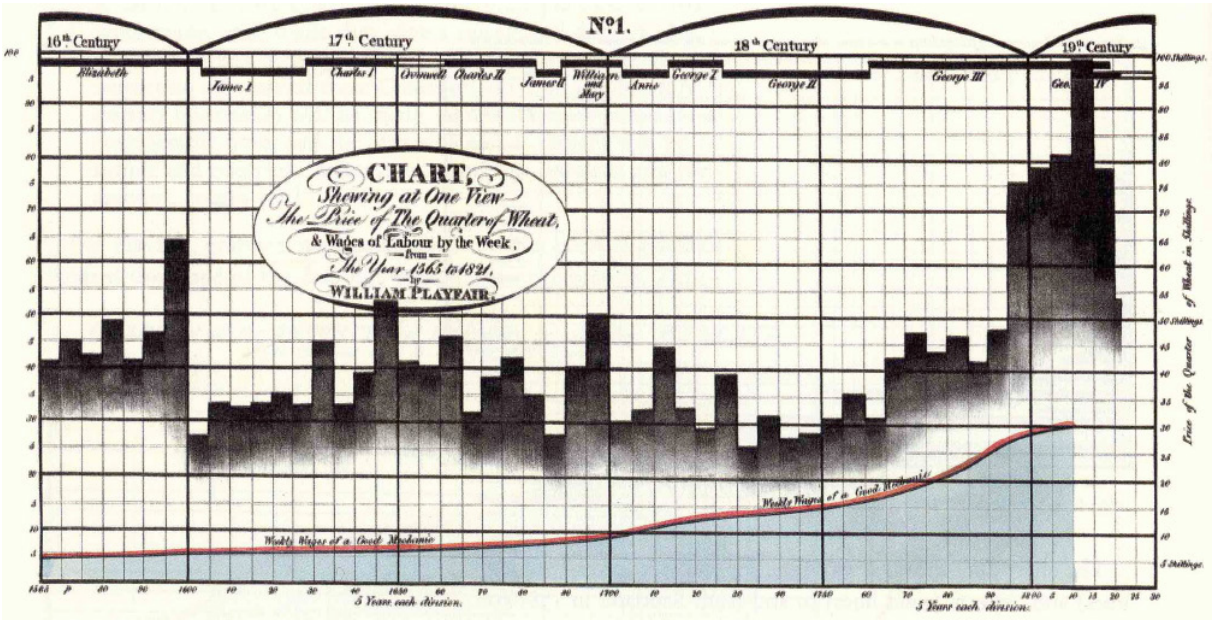



**Fig. 2.5** *Chart, Shewing at One View The Price of The Quarter of Wheat, & Wages of Labour by the Week, from The Year 1565 to 1821* (William Playfair, 1821)
This chart represents three parallel time-series: prices of wheat for 250 years, wages of good mechanics and the reigns of British kings and queens. Through this chart Playfair intended to demonstrate that wheat had never been so cheap as it was at that time, in proportion to mechanical labor.

The first half of the nineteenth century marked the beginning of modern graphics. The number of statistical graphics grew immensely and was accompanied by a great amount of new visual representations of data, such as line and bar graphs, pie charts, histograms, time-series plots and scatter plots. Thematic cartography also kept evolving, advancing from single maps to the creation of comprehensive atlases that featured many types of information. One of such creations that stood out were a series of maps created by André-Michel Guerry (Fig. 2.6), portraying various statistics throughout the various regions of France.

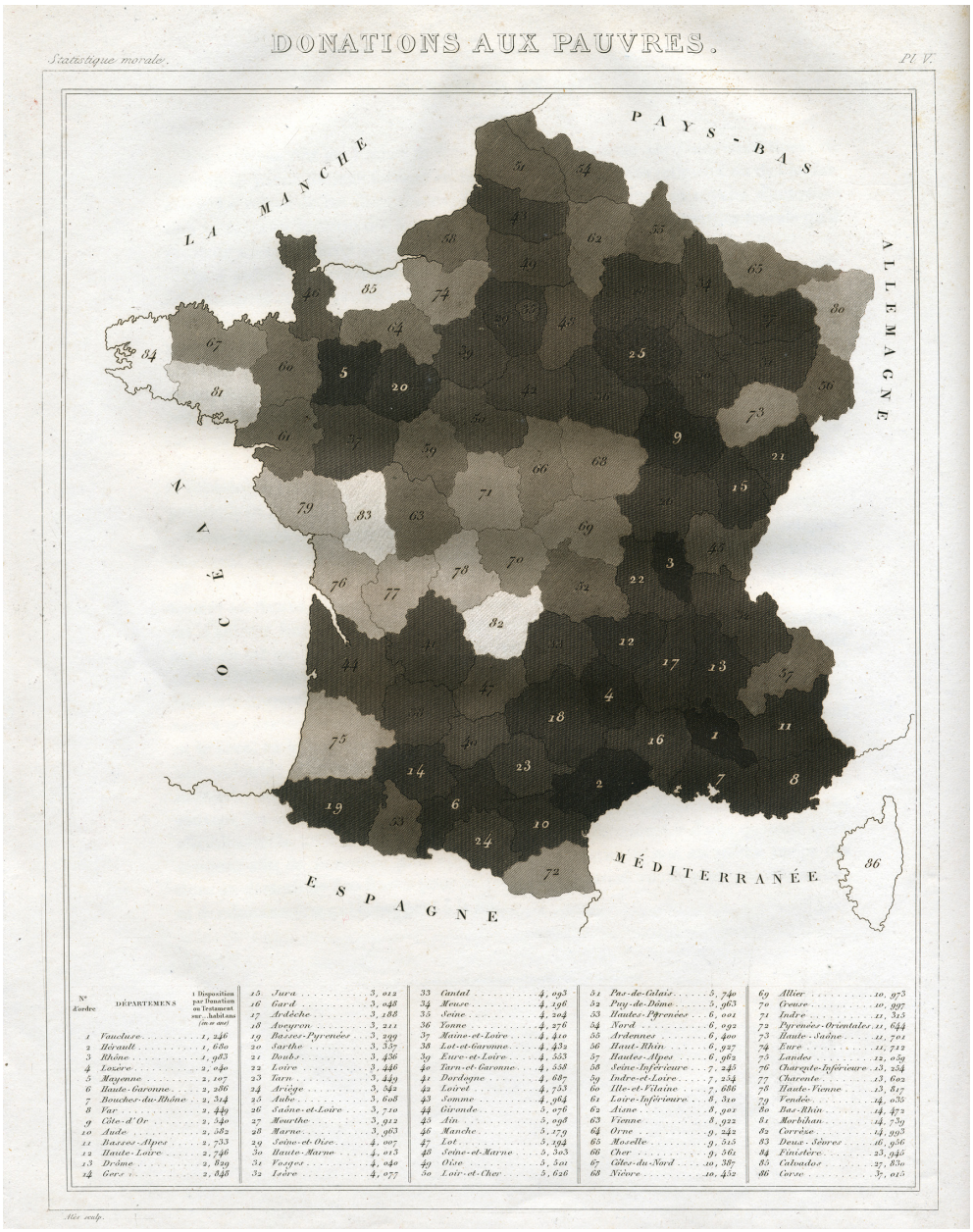



**Fig. 2.6** *Donations aux pauvres* (André-Michel Guerry, 1833)
Map from a series, containing a data table showing the frequency of donations to the poor per number of residents using gradation of shading, where darker shades represents the lower values. It is divided into the eighty-six administrative departments of France, numbered with the growing order of the donations.

The second half of the century was ready for the next step in information visualizations, as the growing importance of information was now being recognized by diverse areas like commerce, industrialization, social planning and transportation. Representations that had already been used earlier in mapping contexts, such as three-dimensional surfaces (Fig. 2.7), were now being used to establish relations beyond that of two variables in an attempt to solve more varied and complex problems, such as John Snow's map of the cholera outbreak in London (Fig. 2.8) which was used to determine the source of this disease, and Florence Nightingale's chart (Fig. 2.9) which was used to bring attention to the extremely poor soldier health conditions, who were dying mainly of preventable diseases.

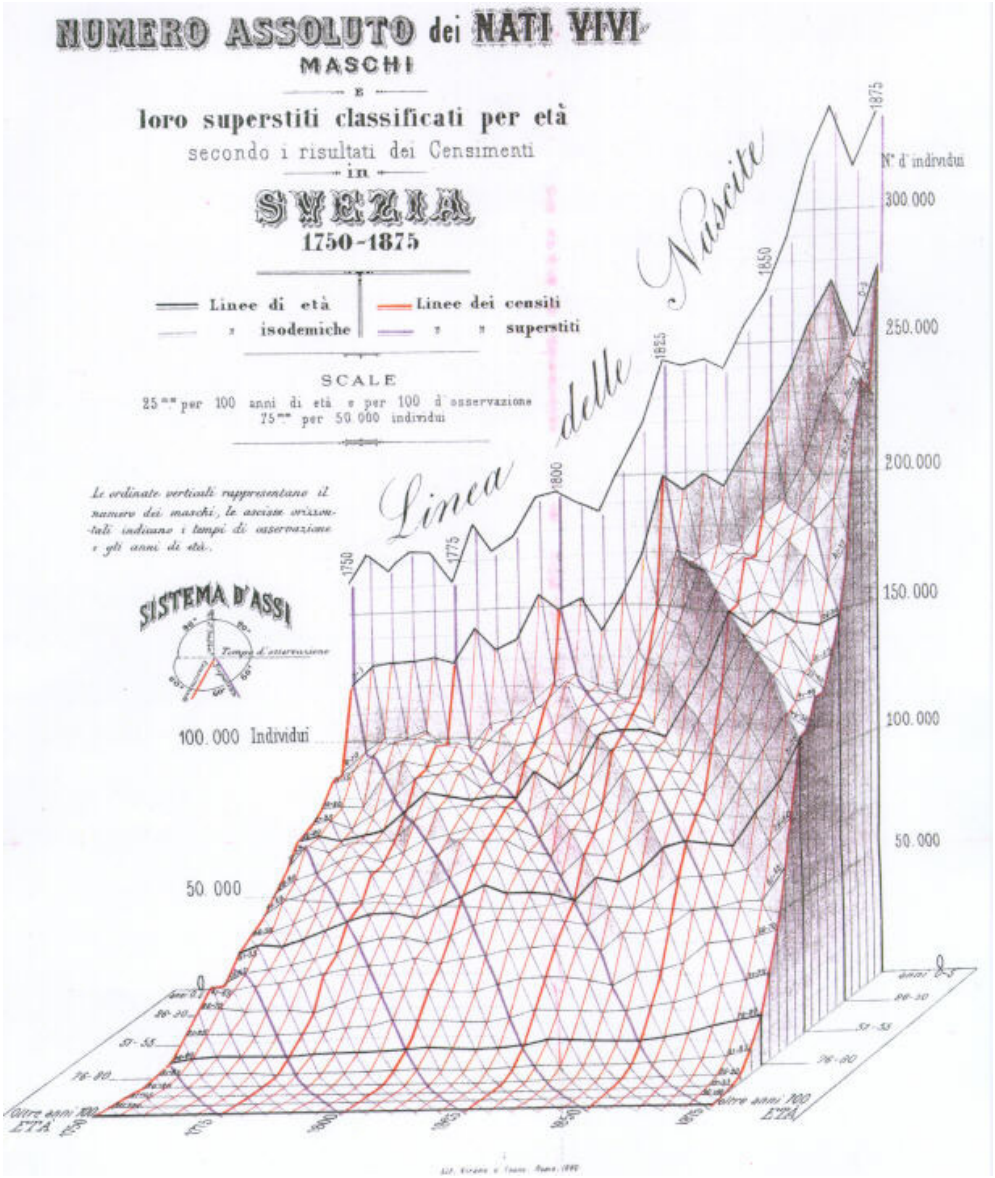



**Fig. 2.7** *Numero Absoluto dei Nati Vivi Maschi e loro superstiti classificati per eta secondo i risultati dei Censimenti in Svezia 1750-1875* (Luigi Perozzo, 1880)
Three-dimentional surface plot of population data from Sweden between 1750–1875, shown using the corresponding year and age group.

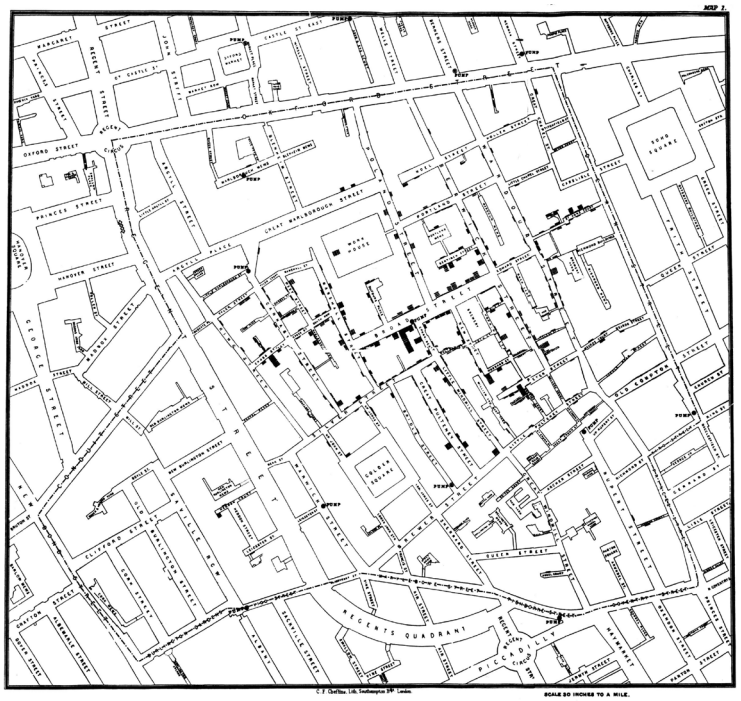


**Fig. 2.8** *Cholera Deaths in Central London* (John Snow, 1854)
Dot map that proved invaluable in the discovery of the cause of the cholera outbreak in central London, found to be the drinking water from the Broad Street water pump. Deaths are represented as short segments and the water pumps are shown as dots.

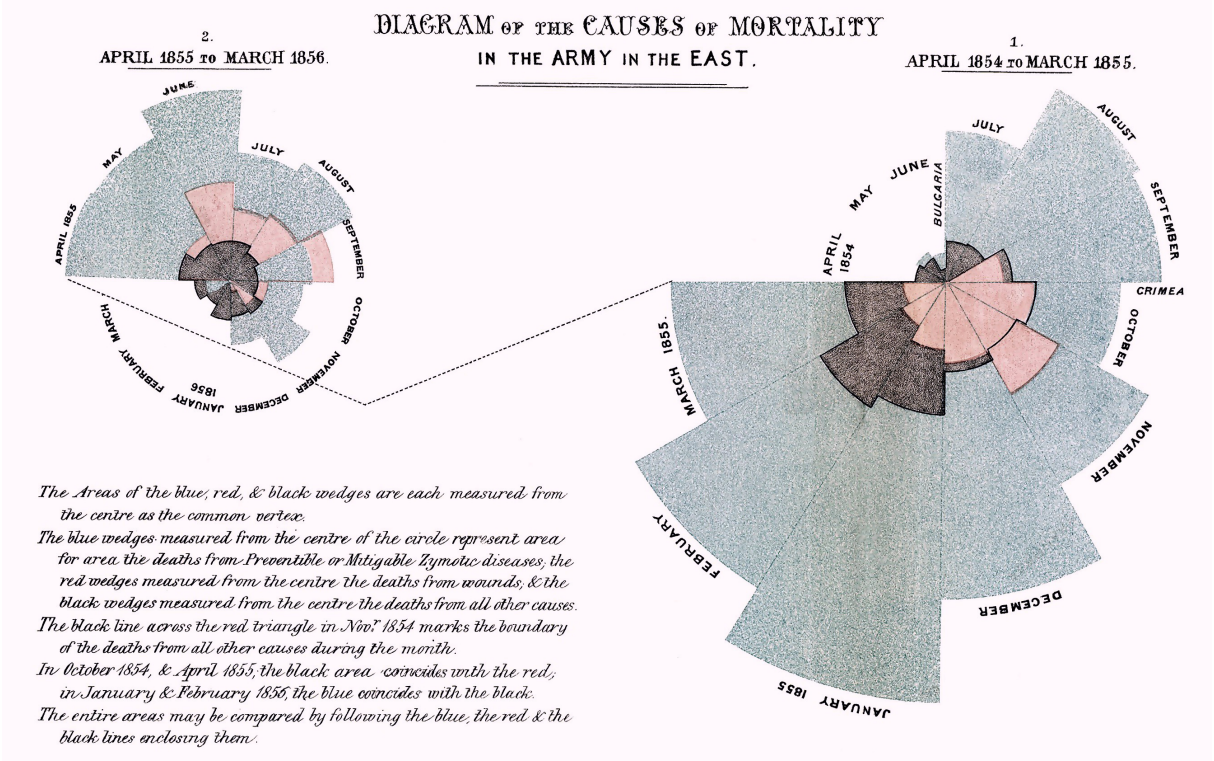



**Fig. 2.9** *Diagram of the Causes of Mortality in the Army in the East* (Florence Nightingale, 1857)
This visualization was the earliest polar area chart, with the purpose of rallying for better and more sanitary treatment of soldiers in the battlefield. Deaths from preventable diseases are represented in blue, deaths from wounds in red, and deaths from other causes are in black. The chart proved to be a solid argument on how many more soldiers died from poor medical conditions rather than by the hands of the enemy.

Charles Joseph Minard (1781-1870) was highly regarded due to his graphical methods and creations in engineering and statistics. Minard developed innovative visual representations like the use of divided circle diagrams and flow lines on maps, and his flow map of Napoleon's Russian campaign of 1812 (Fig. 2.10) was considered to be one of the "best graphic ever produced" by Edward Tufte [2], who also shows that the map exemplifies many of the fundamental principles of analytical design [4].

In opposition to the graphic visualization advancements and copious inventions of the previous century, the beginning of the twentieth century was characterized as a period of dormancy, with few graphical innovations. However, this was also a period of popularization, where graphics and their past developments became mainstream. A notable creation of this period was Henry Beck's redesign of London's Underground (Fig. 2.11), who used his engeneering background to come up with the idea of simplifying the map into something closer to an electrical schemetic.

Towards the middle of the century, the field of data visualization witnessed significant developments such as John Tukey's paper – *The Future of Data Analysis* (1962) [5] – which recognized the importance of data analysis in the field of statistics and differentiated it from mathematical statistics. Another large contribution to the field was made by Jacques Bertin (1918–2010), who organized visual and perceptual graphical elements in accordance to their relations with the data in his book – *Semiologie Graphique* (1967) (which was later published in English [1]).

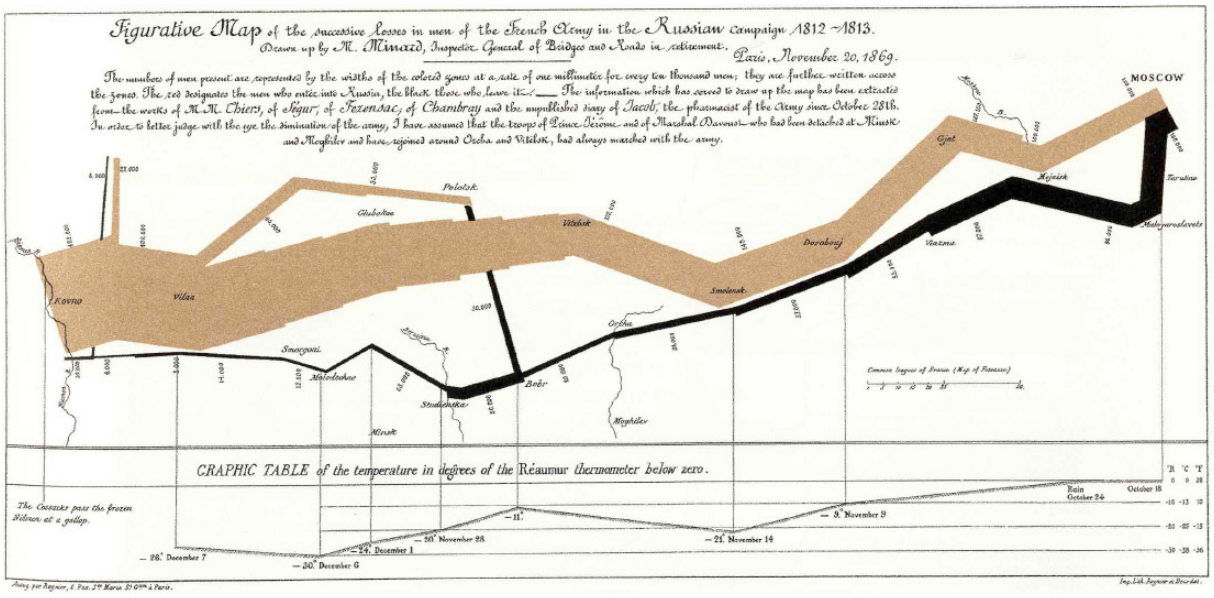



**Fig. 2.10** *Figurative Map of the successive losses in men of the French Army in the Russian Campaign 1812-1813* (Charles Joseph Minard, 1869)
This chart illustrates Napoleon's Russian campaign of 1812, and displays a great variety of information simultaneously. The thickness tan flow-line represents the size of the Grand Army as they progress from the left, starting on the Polish-Russian border, to the right, with Moscow as their destination. The dark flow-line represents Napoleon's retreat, once again with the thickness representing his army's decreasing size. This flow-line is also linked to a temperature scale below with the respective dates.

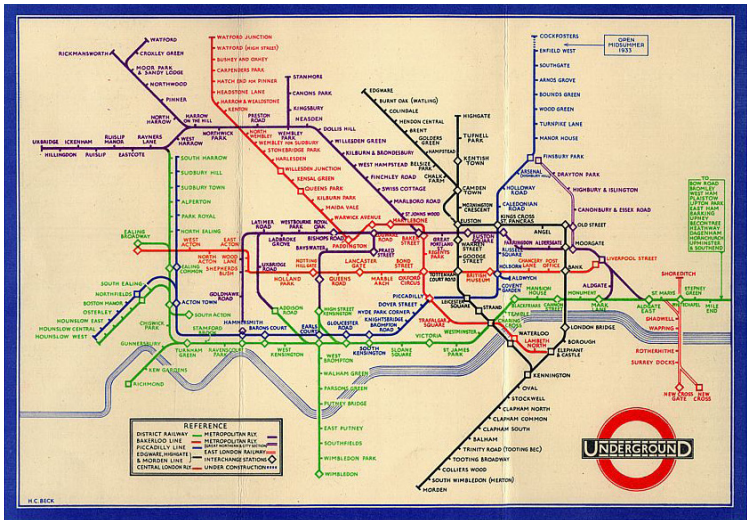



**Fig. 2.11** *London Underground Map* (Harry Beck, 1931)
Harry Beck's redesign of London's Underground simplified the map of the stations into that of an electrical circuit diagram. Though it distorted the geographical positions, it also facilitated the perception of the passenger's desired stations, as they would not require the exact positions when using the underground.

The creation of *FORTRAN* (1957), the first high-level programming language, marked the beginning of the development of computer programs that could process statistical data and construct graphic visualizations. Herman Chernoff took advantage of these rising technologies to represent data creatively by generating faces from inputed data (Fig. 2.12), which had later applications such as Eugene Turner's map that represented the quality of life in Los Angeles (Fig. 2.13).

The twentieth century continued to witness many innovations in various areas during its last quarter. Technological developments such as increased computer processing speed and memory capacity allowed for the processing of large amounts of data and the creation of high-density visualizations. These advancements extended to dynamic visualizations and the development of new interaction paradigms, furthering the possibilities for and efficiency of direct and instantaneous manipulation of the data and graphic proprieties.

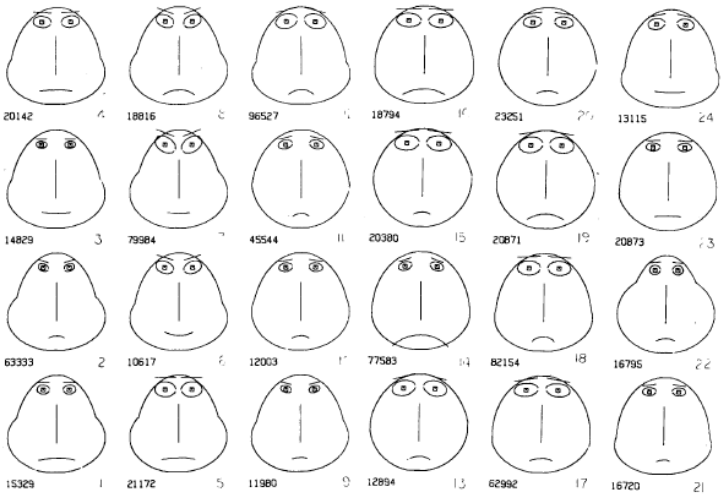



**Fig. 2.12** *Chernoff's faces* (Herman Chernoff, 1973)
The image shows one of the groups of faces that were created by Herman Chernoff. They used to represent multivariate data by representing the values using the various facial features like the eyes, nose, mouth, eyebrows and the head shape, creating figurative facial expressions.

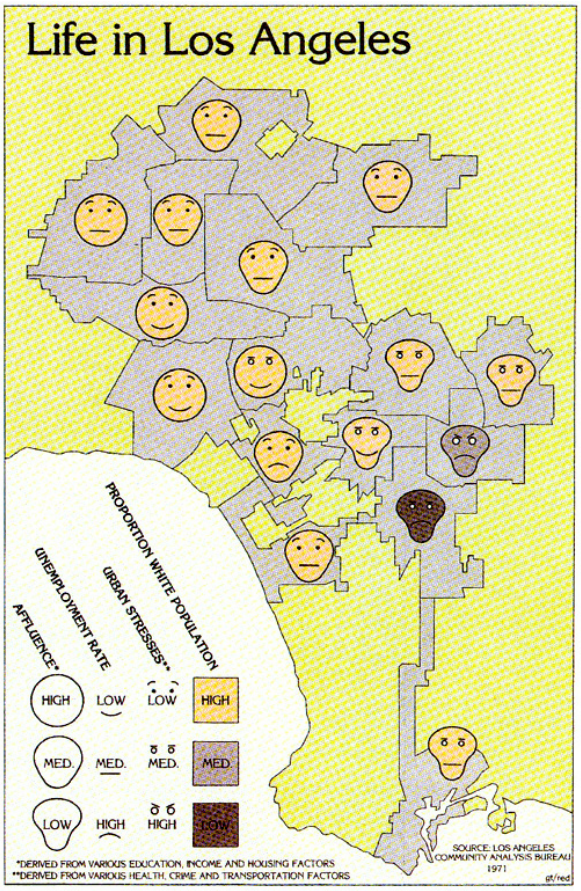



**Fig. 2.13** *Life in Los Angeles* (Eugene Turner, 1977)
This is an application of Chernoff's faces representing the affluence, unemployment rate, urban stress and percentage of white population in Los Angeles by mapping these to facial features. The the expressive facial features are also representative of whether the influence of the values on the quality of life is positive or negative so that they can be conveyed through the appropriate emotions. For example, low unemployment rate is a positive aspect and will create a smiling face.

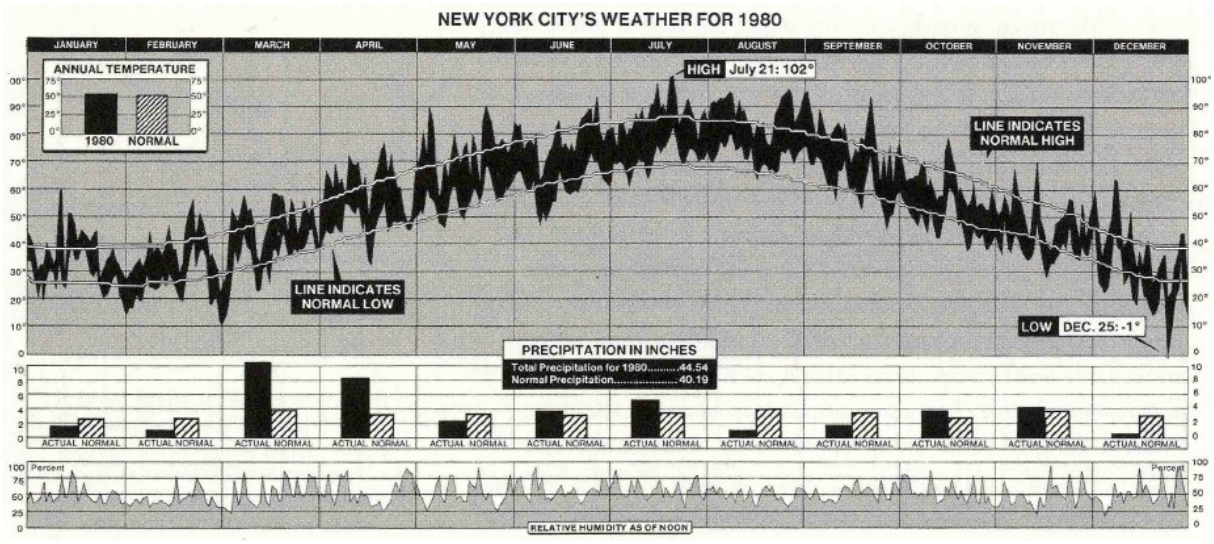



**Fig. 2.14** *New York City's Weather for 1980* (New York Times, January 11th, 1981)
This visualization represents the temperature, precipitation, and relative humidity throughout 1980 in New York City, as well as a long run average and a forecast of the expected change ovr the year. It is an example of a successful attempt at representing a large collection of data (1,888 values) in a way that is easily understood, and tells a story.

The end of the century also accommodated the popularization of data graphics by the mainstream media, such as newspapers (Fig. 2.14), as well as allowing the average public to more easily access existing data. The Internet was one of the major contributors in this aspect as it changed how the world communicated, so it had a natural influence over the field of data visualization, especially with it's popularization in the 1990's.

Social networking became exceedingly popular in the twenty-first century, and with the ability to access the data of these services came the creation of visualizations that would reflect these massive networks of information from people all around the world, such as *Just Landed*, a project created by Jer Thorp using data gathered from the social network *Twitter* (Fig. 2.15), drawing connections between people's home locations and the origin of their tweets which contained the phrase "just landed".

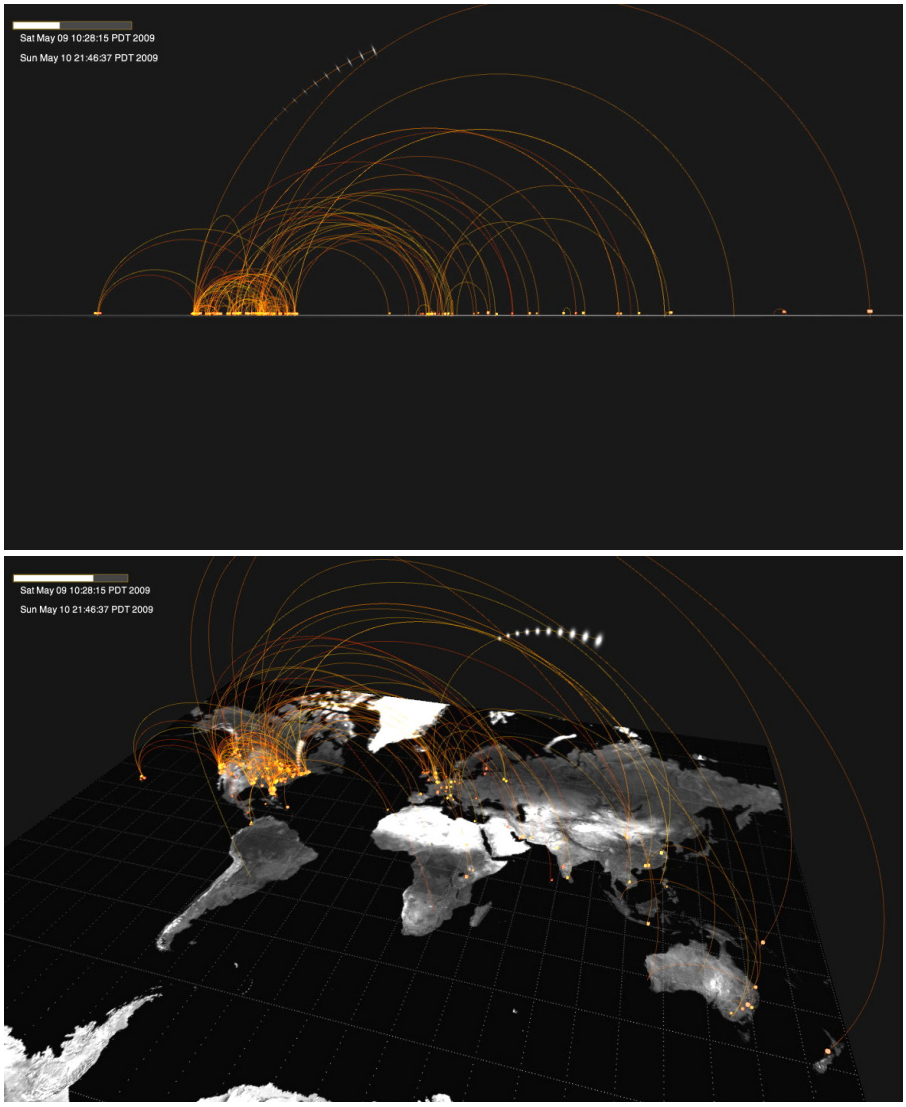



**Fig. 2.15** *Just Landed* (Jer Thorp, 2009)
This visualization predicts and portrays a series of flights with only information gathered from the social network *Twitter*, mapping people that twitted the phrase "Just landed in" on a geographical map by using the poster's home location to create the origin point and reading the message itself to find the destination.

Most notably with the turn of the century was how the new techonologies found its way into the common person's everyday life, which meant most people now had access to the large databases found online as well as software tools which were becoming more mainstream and easy to use. This lead to the emergence of many kinds of visualizations representing a multitude of varied subjects, from science (Fig. 2.16) to religion (Fig. 2.17), the mainstream media (Fig. 2.18), literature (Fig. 2.19), diseases (Fig. 2.20) and even proteins (Fig. 2.21) or attempts to map the internet itself (Fig. 2.22).

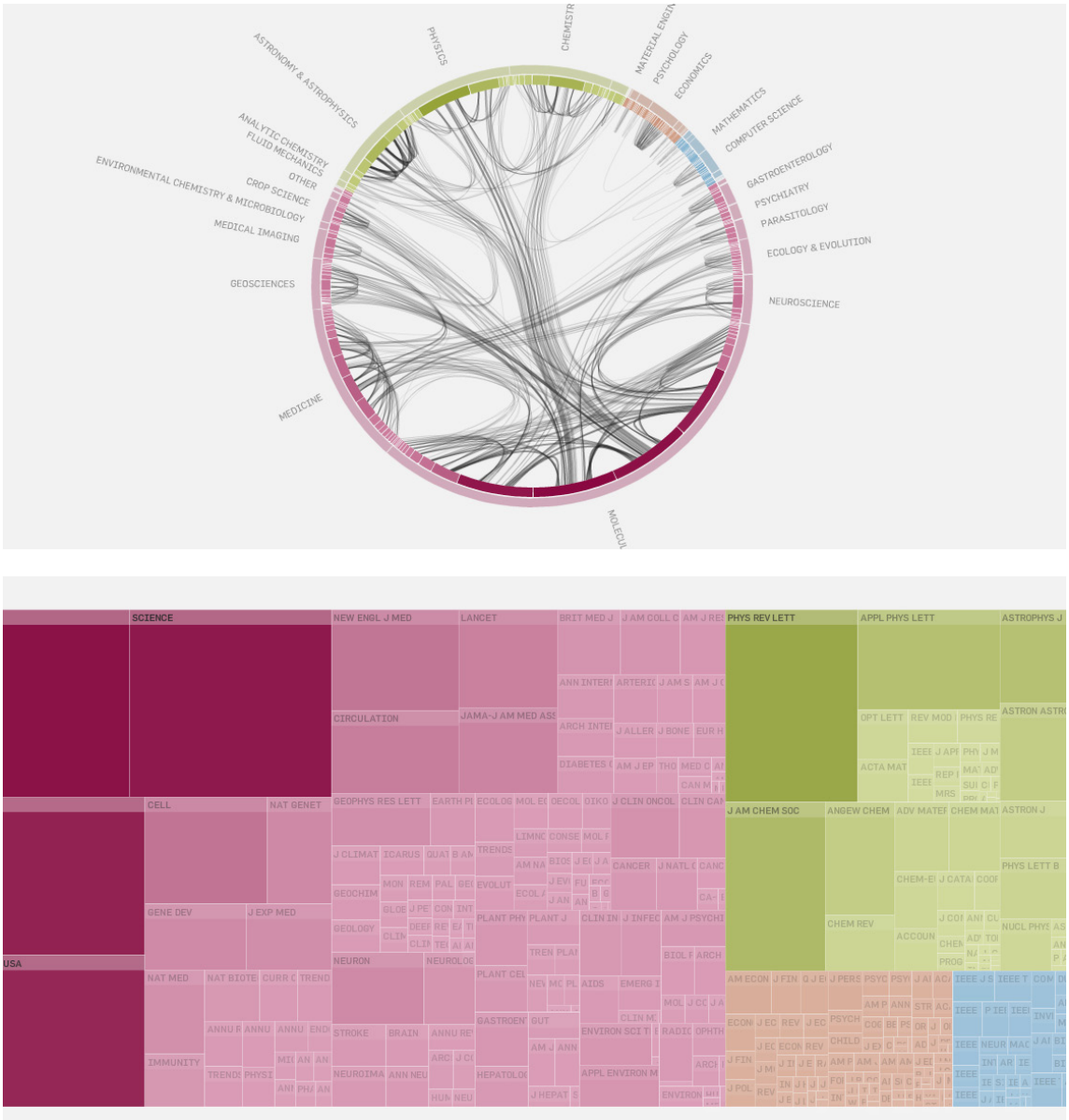



**Fig. 2.16** *Visualizing Information Flow in Science* (Moritz Stefaner, 2009)
Also known as the *Eigenfactor* project, it is composed of set of four visualizations, each with a different structure – radial diagram, map, flow diagram and treemap – representing a calculated factor of importance for individual journals, as well as the in and out citation flows. There are four categories of scientific journals: medical, natural, formal and social sciences. The images above show the radial diagram and tremap.

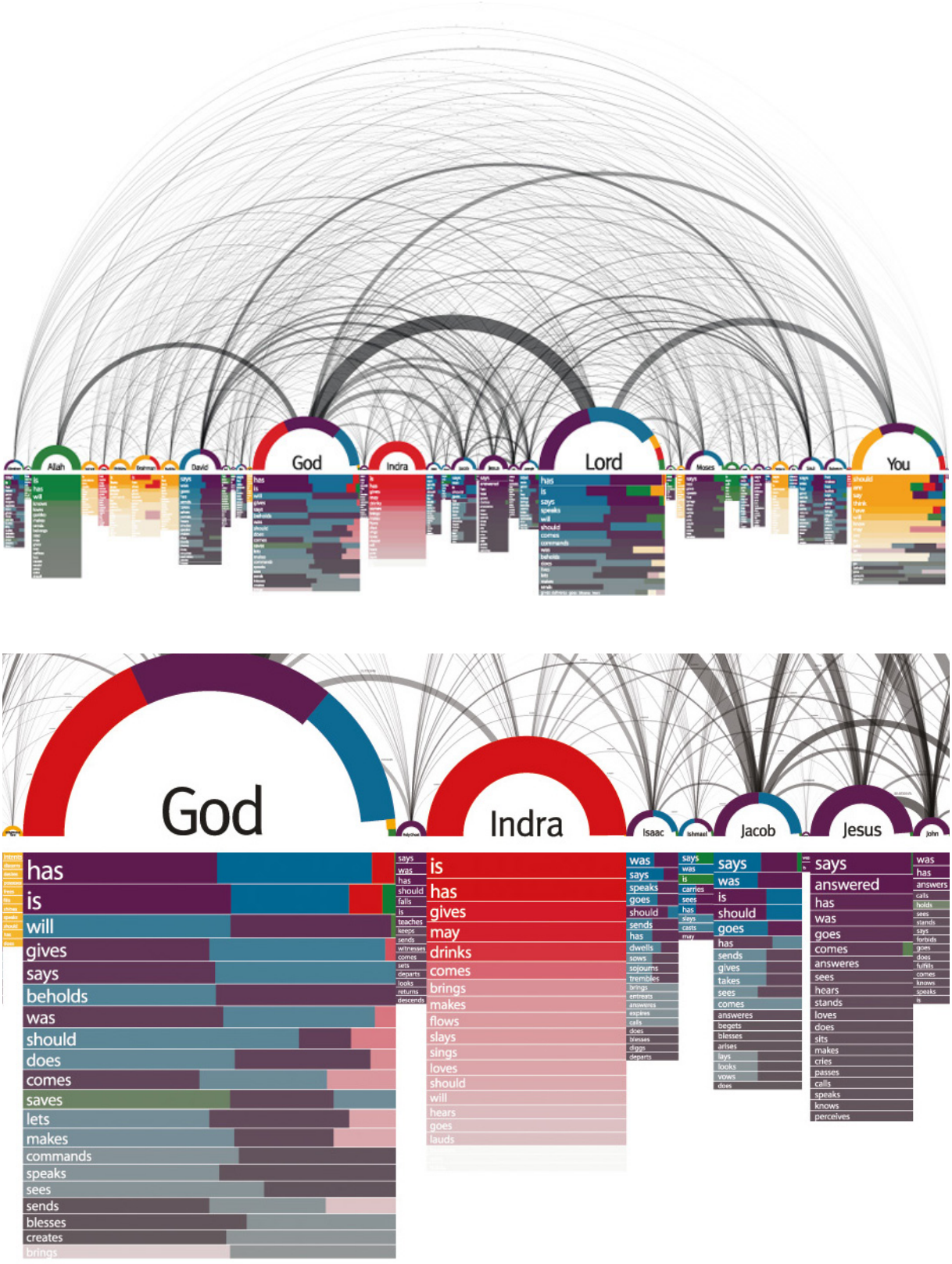



**Fig. 2.17** *Similar Diversity* (Philipp Steinweber, Andreas Koller, 2007)
An arc visualization showing the similarities and differences between the holy books of various religions – Christianity, Islam, Hinduism, Buddhism, and Judaism.

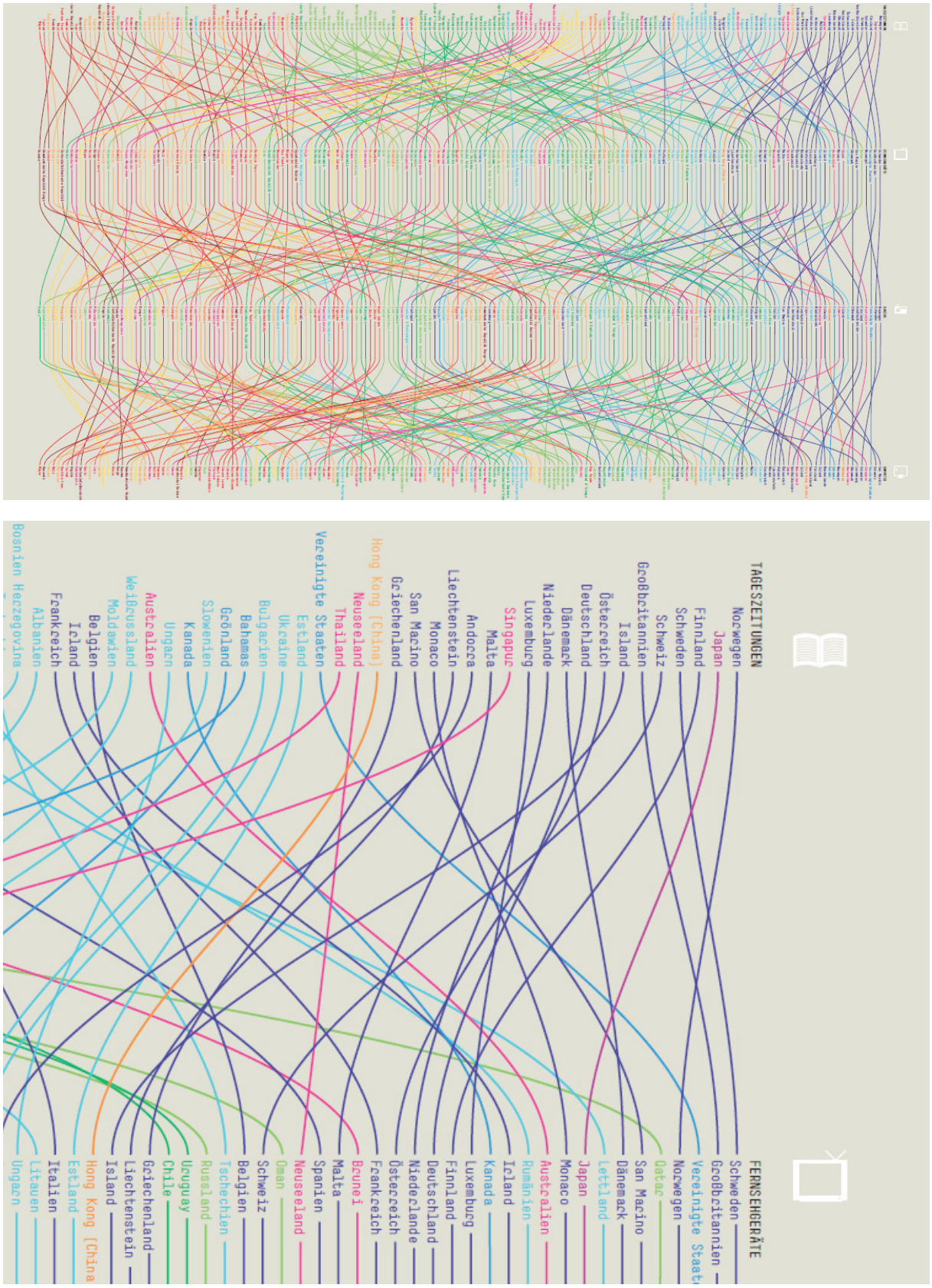



**Fig. 2.18** *Overnewsed but uninformed* (Stefan Brautigam, 2008)
A network comparing various news media outlets (newspaper, television, radio and internet) through different countries, which are simply sorted by rating and then connected in order to visually represent the fluctuations between the media most used in each country.

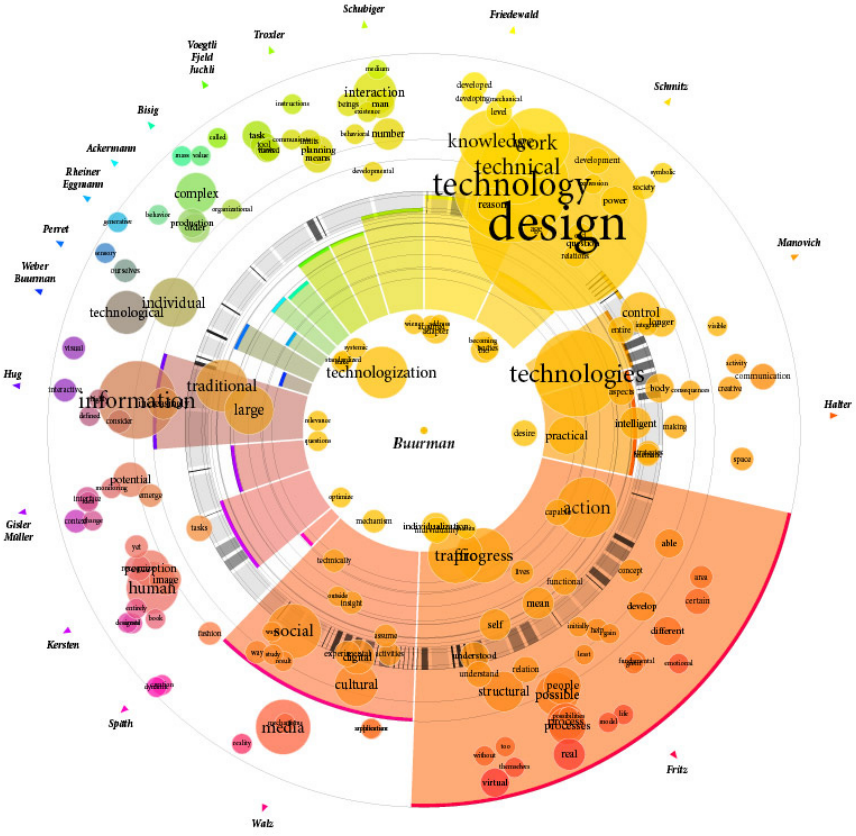



**Fig. 2.19** *Graphical Visualization of Textual Similarities* (Jürgen Späth, Magnus Rembold, 2006) This visualization uses a combination of scaling circles with a polar chart to represent the thematic links between nineteen essays found in the book *Total Interaction: Theory and Practice of a New Paradigm for the Design Disciplines* (2004).

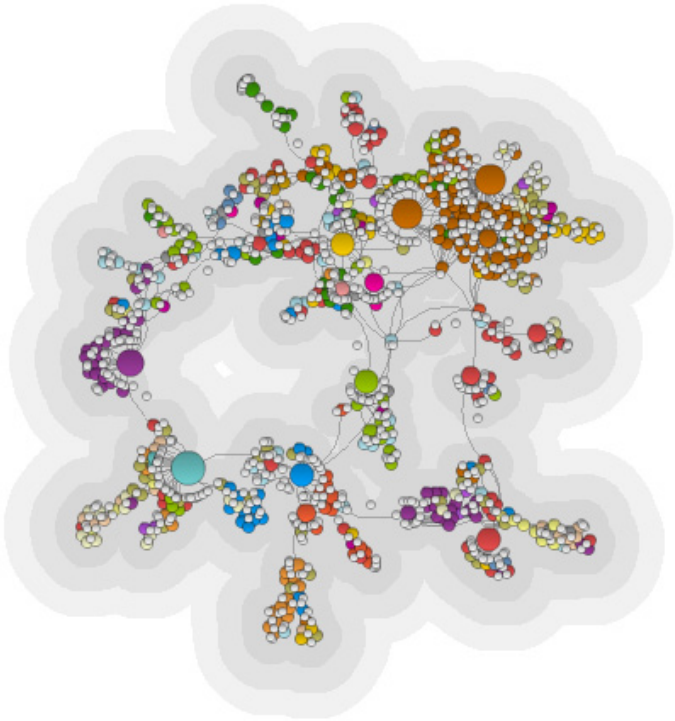
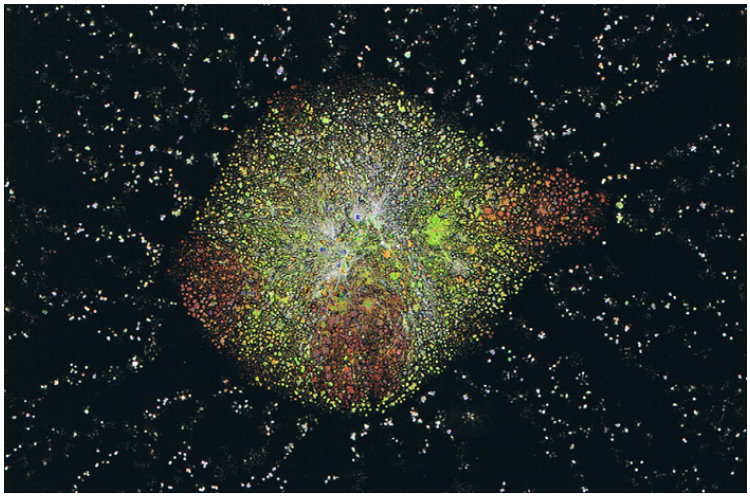


**Fig. 2.20** *Diseasome* (Linkfluence ,2009) An interactive map of the human-disease network, consituted by 903 genes, represented by white nodes, and 516 diseases, represented by the remaining colored nodes, divided across 22 categories. Clustering is made more evident throught the use of colored borders, which emphasises the genes and diseases with more correlations between each other.



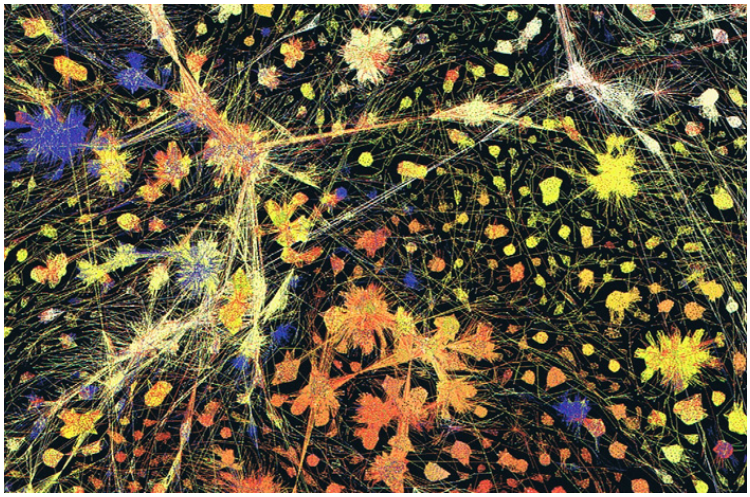


**Fig. 2.21** *Minimum Spanning Protein Homology Tree*
(Alex Adai, Edward Marcotte, 2002)
A high-density visualization of a protein-homology map composed of over 300,000 proteins. Edges are red if connecting two proteins of the same species, red if they are different, or a color based on the layout hierarchy if it's unknown. Proteins have specific colors based on a classification system determined by their genetic lineages.

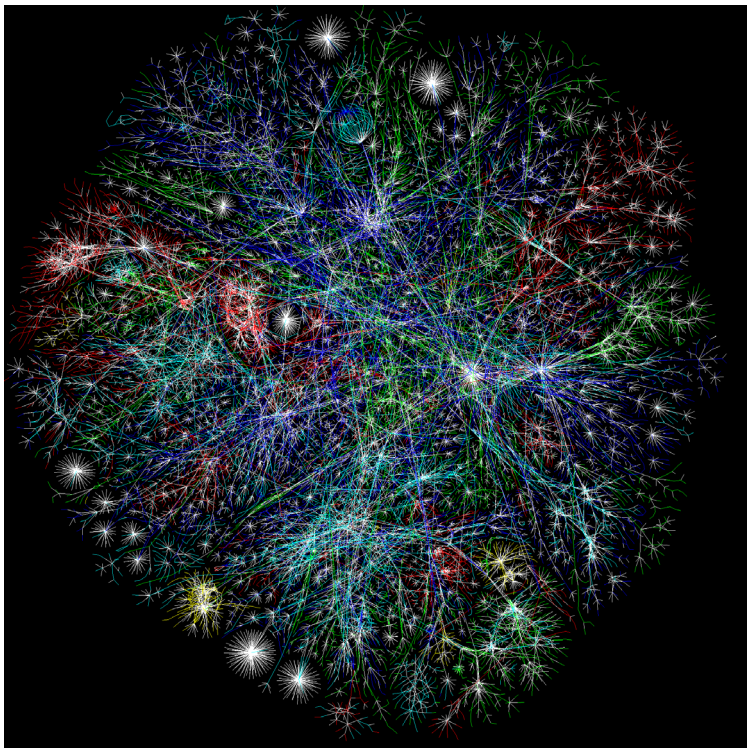


**Fig. 2.22** *Opte Project* (Barrett Lyon, 2003)
The Opte Project was an attempt to create a visual representation of the internet. The image shows a high-density network representing the internet map from November 23 2003, with over 5 million links from various IP addresses. It can be used to visualize the overall growth of the Internet, as well as how specific areas change over time.

# Semiology of graphics

Jacques Bertin was a French cartographer and theorist, known for his monumental work in the field of information visualization. His book entitled *Semiologie Graphique* (*Semiology of Graphics*) [1], released in 1967, was composed of the most complete and elaborate studies on the theory of information visualization and was one of the most far-reaching efforts to provide a theoretical foundation to the field.

Bertin's work presented a new direction in the primarily subjective field of graphic representation and information design where he attempts to summarize the principles of graphical communication into simple logical rules that help both restructure the available data and find the best way to represent it visually so that it can easily be understood. The understanding of his work proves to be an essential stepping stone in the creation of any information visualization as his methods allow for a simple understanding of the basic elements that make up a visualization and dictate the working relationships with the various types of data we wish to represent.

*"Graphics owes its special significance to its double function as a storage mechanism and a research instrument. A rational and efficient tool when the proprieties of visual perception are completely utilized, graphics is one of the major "languages" applicable to information processing."*

– Jacques Bertin, 1967, Semiology of Graphics [1], p.2

Bertin identifies the graphic system of a visualization as being made up of two parts: the *content* and the *container.* The content consists of all the information that will be represented, while the container represents the proprieties of the representations, namely the graphic system.

The process of creating a visualization starts with the organization of the content, which requires the analysis and categorization of all the information available. As Bertin states, the first step in analysis the information is identifying the *invariant*, which is the main idea which will be transmitted through the *components*. These components are the variational concepts which need to be identified through the data. The next step is establishing the *elements* or *categories*, the identifiable parts of component, as well as determining the *length*, which will define the number of elements in a component.

The visual representation of the elements requires marks that in order to be visible must reflect light which is different from paper, such as ink or, in a more modern context, the pixels of a screen. These marks can vary in position and in the way they are used, to which Bertin [1] assigns the name of implantation. On a plain, the marks can be defined as points, lines or areas. The visual proprieties of the marks are named *visual variables* and a mark can express a correspondence between the two planar dimensions through its position. When fixed at a certain point, a mark can also vary in size, shape, orientation, color and texture. These are the visual variables established by Bertin at the time, although there have some proposed additions throughout the years (Fig. 2.23).

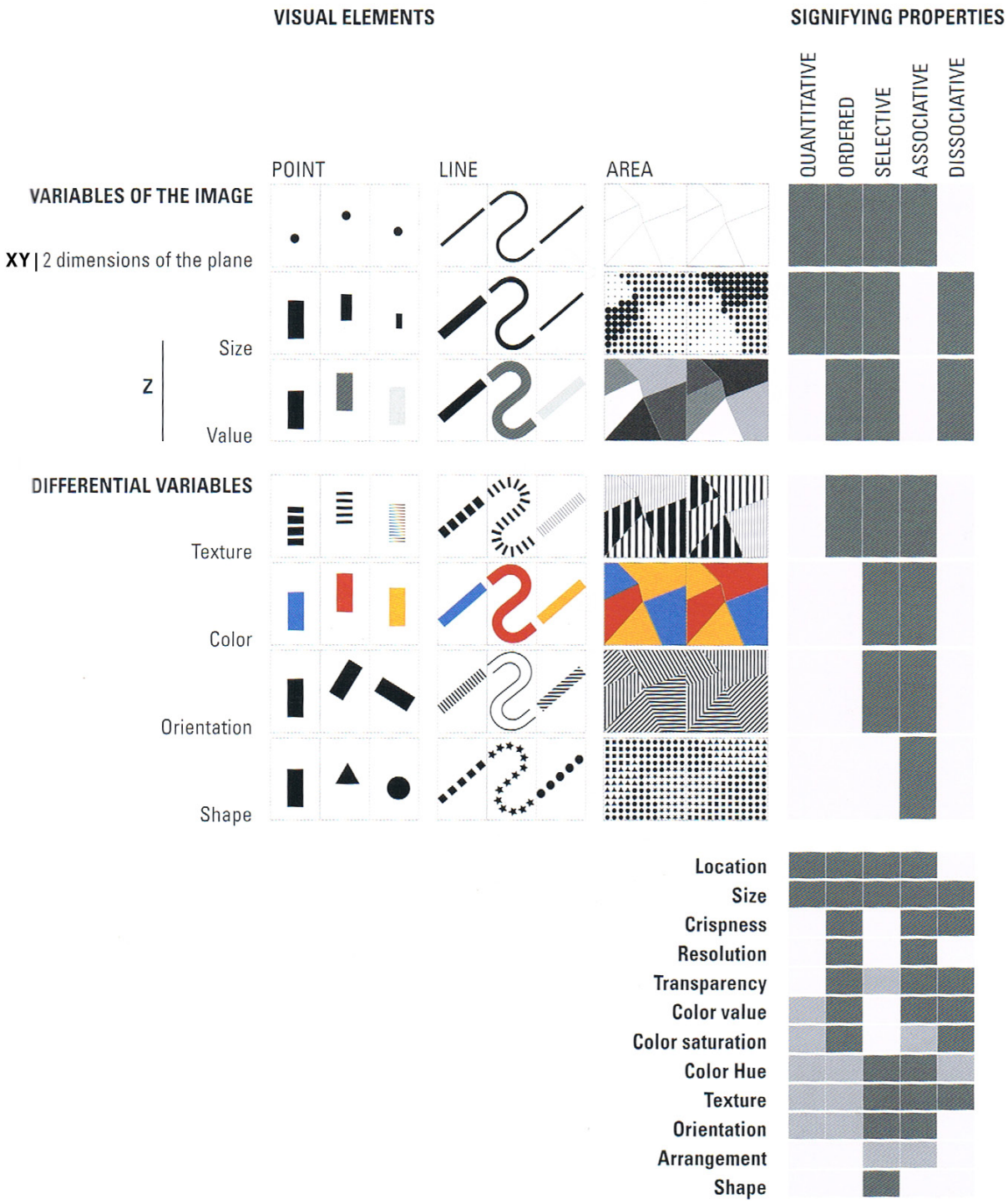



**Fig. 2.23** This chart, created by Isabel Meirelles [6], presents Bertin's system of perceptual variables, with the smaller chart on the right indicating the appropriateness of each variable to its level of organization. The bottom-right chart is a more recent expansion that includes other variables that have been added over time; in this chart the lighter grey signifies "marginally effective". The chart also includes clarity, a new visual variable introduced by Alan MacEachren [7], which consists of three listed variables: crispness, resolution and transparency.

Certain variables are more appropriate to represent certain components which depends on the *level of organization* of each component, determined by the analysis of the relationships between each of them. The first level is named *qualitative* or *nominal*, but it's divided into two sub-levels of organization. One is the *associative* level which consists in grouping similar elements, which are better represented through color, texture, shape and orientation. However, size and values are dissociative and can break uniformization. The other level is *selective*, based on the differentiation of the elements. Differentiating can be represented through value, size, texture and color, but shape and orientation (when represented using an area) are not easily percepted. A variable is *ordered* when the represented components have a single and universal order. Texture, value and size are variables with a perceptible order, but color, shape and orientation are not ordered. Finally, the *quantitative* level is used to describe components that have countable units. Only variations in size are appropriate to represent relationships on this level adequately.

The use of planar dimensions is called *imposition*, in which Bertin distinguishes four groups which are defined by the representation of the elements on the plain: *Diagrams* are made up of the correspondence between all the divisions of a component with all the divisions of another, where each component is mapped to an axis of the plain. *Networks* establish relationships between all the divisions of the same component. *Maps* are the same as networks, but their elements are presented in geographical order. *Symbols* do not establish correspondences within the representation, but instead they create relationships between the marks and the reader himself. Their meanings are dependent on the cultural context and knowledge of the person. Diagrams and networks can have different types of imposition based on how the variables are inscribed onto the plain. The imposition can be linear, circular, orthogonal or polar.

# Graphic efficiency

The efficiency of statistical graphics is determined by their communication of complex ideas with clarity and precision. The representation of information should follow certain principles in order to guarantee graphical excellence such as encouraging comparisons between the data, avoiding distorting the data or any manipulations that serve only to favor the aesthetics, representing several levels of detail (such as an overview and a close-up detailed view), and integrating the statistical and verbal descriptions of the data with the visual display.

*"Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space."*

– Edward Tufte, 2001. The Visual Display of Quantitive Information [2], p. 51

Graphics have the power to communicate great amounts and complex information, but the inappropriate representation of the data for the purpose of simply decorating some numbers can only guarantee mediocre graphics.

Edward Tufte heavily criticizes inaccuracies in graphics, holding graphic designers on the same level of integrity of a writer who should not alter the graphical aspect or even the data itself so that these fit into their "aesthetically pleasing" graphic representation. He lists some principles that should be followed in order to retain graphical integrity such as not quoting data out of context, using clear and detailed labels to avoid misinterpretations from graphical ambiguity or distortion, and representing numbers directly proportional to the numerical quantities represented. He also suggests the calculation of a graph's *lie factor*, which represents the ratio between the size of the impact in the graphic be the size of the impact in the data, and if this ratio is above 1, then the visualization is exaggerating the representation (Fig. 2.24).

In order to assure graphical competence the art must work side-by-side with the substantive and quantitative expertise, as statistical integrity must not be sacrificed for aesthetics, though this also does not mean that accuracy demands an ugly graphic. It is by upholding proper standards and through the cooperation of both sides that one can achieve true graphical excellence.


**Fig. 2.24** This chart from the Russian journal Pravda is a simple but clear example where the direction is clear, but the magnitude of the numbers is clearly incorrect, as their representations do not accurately reflect the increasing proportions of the quantities.

Tufte also asserts a few more principles regarding good graphical representations, such as the maximization of the *data-ink* ratio – the proportion of the graphic which contains and transmits actual information –, the avoidance of *chartjunk* – excessive and unnecessary graphical elements, like heavy grids and moiré vibration – and the maximization of *data density,* within reason – high-density graphs can portray a lot of information at once, but legibility should not be sacrificed.

Aside from graphical representations, it is also necessary to have a basic understanding of certain principles about visual perception in order to properly display information. Most of these principles can appear as basic notions, as they tend to be natural reactions even when looking at a simple chart, for example, how we group objects based on them being the same color. However, it is possible misjudge the importance of a visual propriety over another, or to simply have certain elements lead to misinterpretations. One series of principles that describe these perception phenomenons is called the *Gestalt laws*.

The Gestalt laws describe the way on how we detect patterns and how they can be used to effectively enhance perceptual inferences and problem solving, as stated by Isabel Meirelles in *Design for Information* [6]. The laws are simple principles that allow for the visualization to become more coherent and patterns to be more easily perceptible. These principles are defined as proximity, similarity, common fate, good continuation, closure, and segregation between figure and ground. *Proximity* describes the tendency to perceptually group elements that are closer together. This also applies in abstract domains, as graphical proximity corresponds to conceptual proximity. *Similarity* serves mainly for categorical association, as it describes the tendency to group visual elements with similar characteristics that don't relate to location, such as color and shape. *Common fate* is the tendency to group elements with similar directions, or more specifically, moving in the same direction. *Good continuation* is the use of uninterrupted and smooth elements or lines that are straight or smoothly curved in order for representations and, more specifically, the relationships to be easier to read and interpret. *Closure* describes our interpretation of closed elements as a whole which defines boundaries or a shape, applying even to separate units where our mind can fill in small gaps and close the visual element. The *segregation between figure and ground principle* describes the process of organizing the visual elements by selecting some elements as figures and the rest as ground.

# Relational structures

Information visualization is a very broad and extensive subject and a detailed analysis of every type of possible method for data representation is outside the scope of this dissertation. Instead we will focus on the structures which we created for the project, the main of these being relational structures. The focus of relational structures, also known as networks, lies within the patterns of the connections between its elements, not within the elements themselves, in other words, it's the analysis of the relationships in these interconnected systems that lets us perceive new types of information and conclusion from these interdependencies.

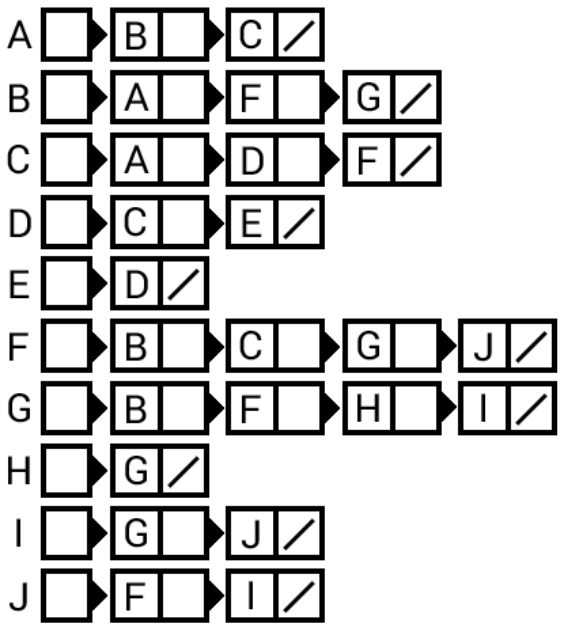
Manuel Lima asserts in *Visual Complexity: Mapping Patterns of Information* [8] that the purpose of network visualization is driven by five key functions: *document* – portray and save new structures; *clarify* – make information more understandable so it becomes easier to process and communicate; *reveal* – find hidden patterns and relationships; *expand* – set the stage for further exploration, and creating an initial outline for future evolution; *abstract* – represent intangible and metaphorical concepts. Bertin defines a network as *"when correspondences on a plane can be established among all the elements of the same component"* (Semiology of Graphics, p.269 [1]). A network can be characterized as a system that's been simplified into an abstract structure composed of edges and vertices, respectively named nodes and links, which can also be labeled with additional information. The elements can be placed on the plane before figuring out their arrangement, and this arrangement should be in a way that produces the minimum number of intersections in order to create and efficient graphic. Leonhard Euler (1707–1783) was one of the first to use networks alongside graph theory and mathematics when solving problems, which was a puzzle involving the city of Königsberg, nowadays Kaliningrad, the former capital of East Prussia, that asked if it was possible to cross all of its seven bridges without crossing the same bridge twice. In 1736, Euler proved that the path didn't exist by converting it into a graph and analyzing the nodes and links, identifying the conditions of what was to be called an Eulerian path – a trail that passes through each node only once.

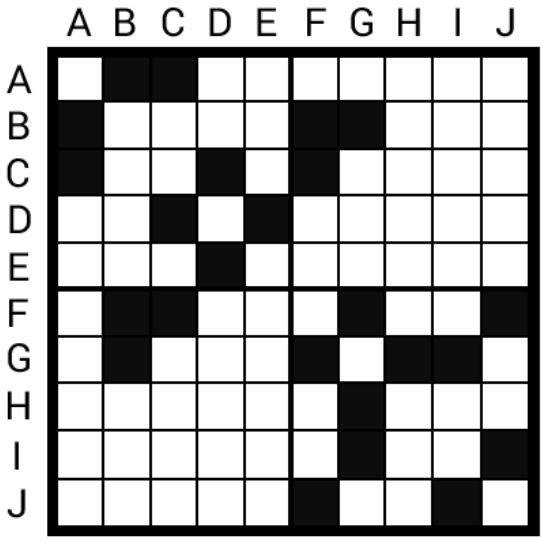
If all the nodes in a network are of one type, it is called one-mode or unimodal, otherwise the network will be called multimodal. There are also networks that consist of two sets of nodes that only share links between sets, and not between each other, in which case they are named bipartie or two-mode. The attributes of links can describe the direction of the interaction – *directed* or *undirected* – and the weight of the connection – *weighted* or *unweighted*. Directed links, also known as asymmetrical edges, have a known origin and destination, making it possible to establish a direction, while undirected links, or symmetrical edges, refer to mutual connections and have no origin or destination attributes. Weighted links represent additional information about a connection, such as its strength. Unweighted links only present whether a connection exists or not. The number of connections of a node is named *degree*, and in directed networks we can distinguish the number of links destined to that node as *in degree* from the number of connections

originated from that node called *out degree*. The analysis of the degree in each of the nodes in a network can help determine certain proprieties, such as centrality.

There are three main types of structures used to describe or represent networks: *adjacency lists*, *adjacency matrices* and *node-link diagrams* (Fig. 2.25). An adjacency list is simply composed of a vertical list of all the nodes in a network, where each node is followed by a horizonal list of all it's links. They are not used often due to being unmanageable and confusing when dealing with large networks. Adjacency matrices are comprised of a grid of nodes, where the cells simply represent whether or not a connection exists, usually through a two-color scheme, and they have the advantage of avoiding the overlapping information that can cause confusion in complex networks represented through node-link diagrams, though they are more visually constricted. Weighted networks require a more complex representation in order to display any additional information regarding the proprieties of the links. In node-link diagrams, nodes are usually represented through symbolic elements and lines are used to represent links, though lines can represent both nodes and links, and so can areas if the type of relationship is inclusive. The orientation of a link can be represented visually, with an arrow for example, but other components must be represented with visual variables. Node link diagrams are usually the most common since they allow for the most freedom. Some common node-link diagram network layouts are the *linear*, *force directed* and *circular* layouts. Linear layouts are where nodes are displayed on a line and links are usually represented with arcs, which, on their own, are usually more appropriate for representing less complex ideas. Force directed layouts use algorithms that simulate physical forces in order to position nodes, making it one of the better layouts for identifying clusters. Circular layouts have nodes organized in a circle, usually grouped by categories, while links are displayed inside the circle (Fig. 2.29).

However, there are many other types of networks as these can vary in a many aspects to create completely different visualizations. Regarding the data itself, the network can be classified differently depending on the distribution of the relationships, as illustrated by Paul Baran in Fig. 2.26. A network is *centralized* if there is a "central node" that all other nodes connect to (a high degree), *decentralized* if there exist various "central nodes" with a large number of individual connections, or *distributed* when the degree value of the nodes is more or less constant throughout all of them. Networks may also vary through the aspects of representation itself, from the disposition of the nodes (which can be displayed in a line or a circle, or multiple lines or circles, or organized using forces, or other methods) to representation of the nodes and links (which can be done through simple or complex shapes or lines, which can then also vary in color, size). There exist an immense number of combinations and possibilities, and in this dissertation we included two lists which document various types of networks grouped by different parameters, one by Jacques Bertin (Fig. 2.27) showcasing some networks across different types of implementations, and another by Manuel Lima (Fig. 2.28) showing some of the possibilities between changing the position and representation of nodes and links.



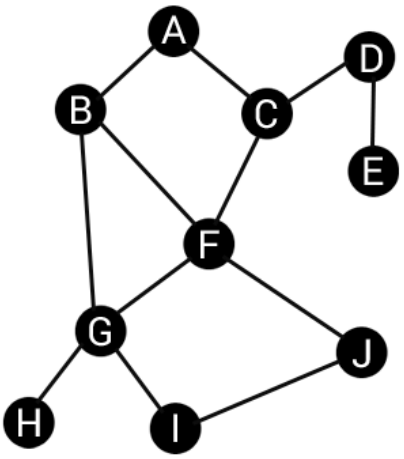




**Fig. 2.25** These three images show the same network being represented using different structures: an adjacency list (top left), an adjacency matrix (top right), and a node link diagram (bottom right).

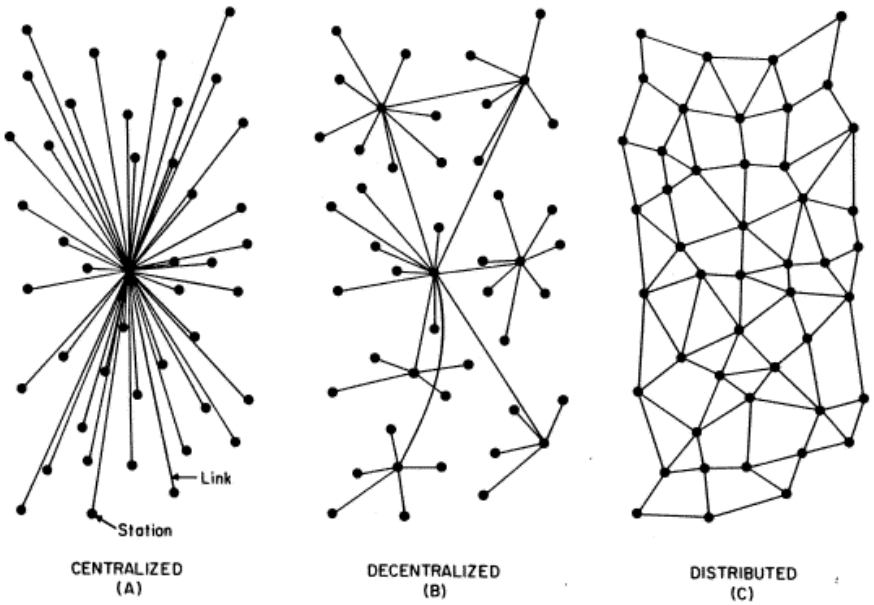



**Fig. 2.26** *Network Models* (Paul Baran, 1964)
A simple representation of three types of networks based on the distribution of the node-link relationships. From left to right, these models are named centralized, decentralized, and distributed.

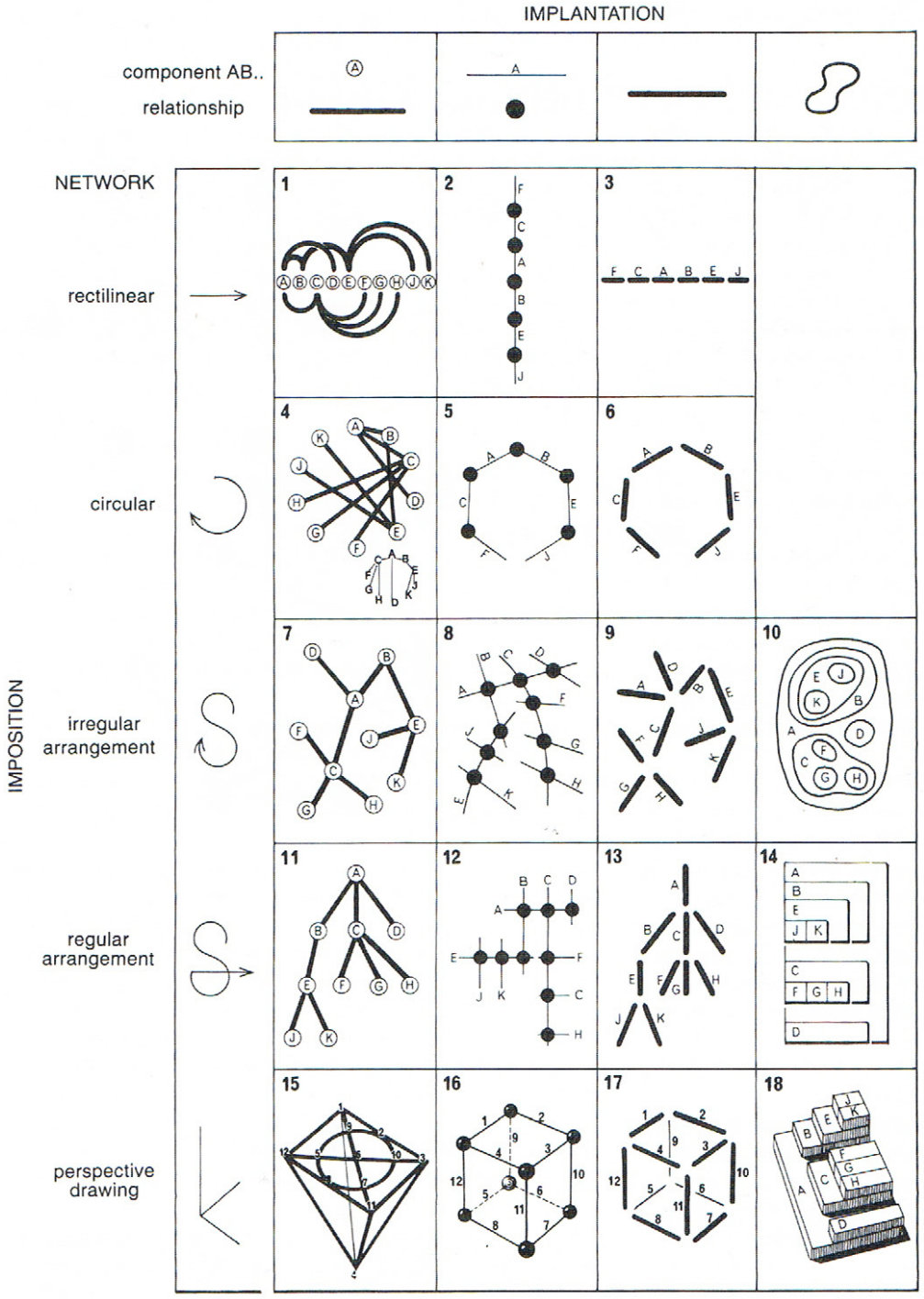



**Fig. 2.27** This chart was created by Jaques Bertin [1], where he distinguishes the different types of network constructions across the various types of implantations.

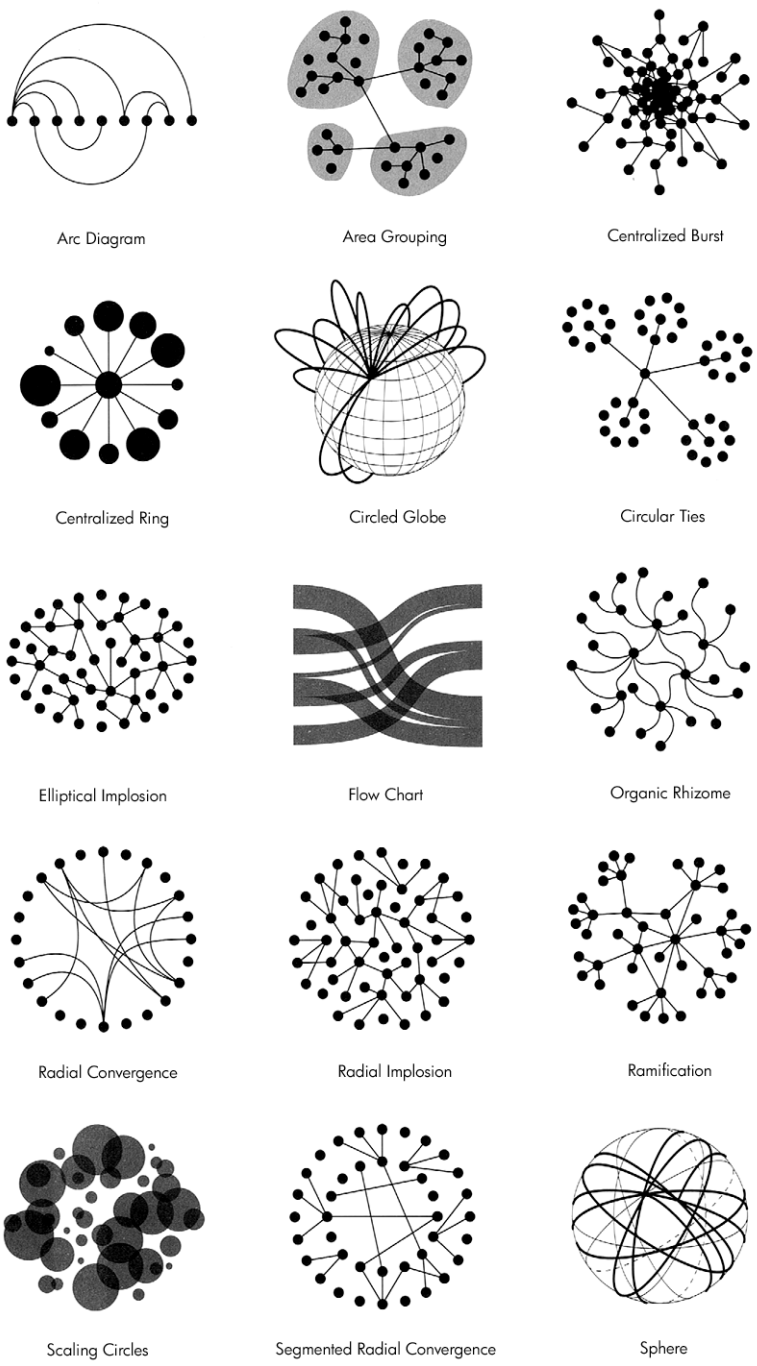



**Fig. 2.28** Another chart that distinguishes various types of networks, based on both the representation and the organization of their nodes and links. It was created by Manuel Lima for his book, *Visual Complexity* [8].

The occlusion of the structures caused by the nodes and link crossings is a well-known problem, having been the target of new layout techniques and algorithms that aim to maximize the legibility of the graph, such as *edge--bundling* (Fig. 2.30), where the links of the network are bundled together in order to make their similarity and directions more perceptible –, or *focus + context* – which is more commonly used in maps, where the amount of detail adapts to the scale (Fig. 2.31). Other strategies try to reduce the number of nodes using *clustering*, where nodes are grouped based on similarity, like categories (Fig. 2.29), or by adding filters to the data (which is also shown on Fig. 2.31), where the three buttons on the center-left allow to switch between types of data. These type of functions are very useful for displaying and helping the user navigate through large datasets, even when they are not necessarily networks.

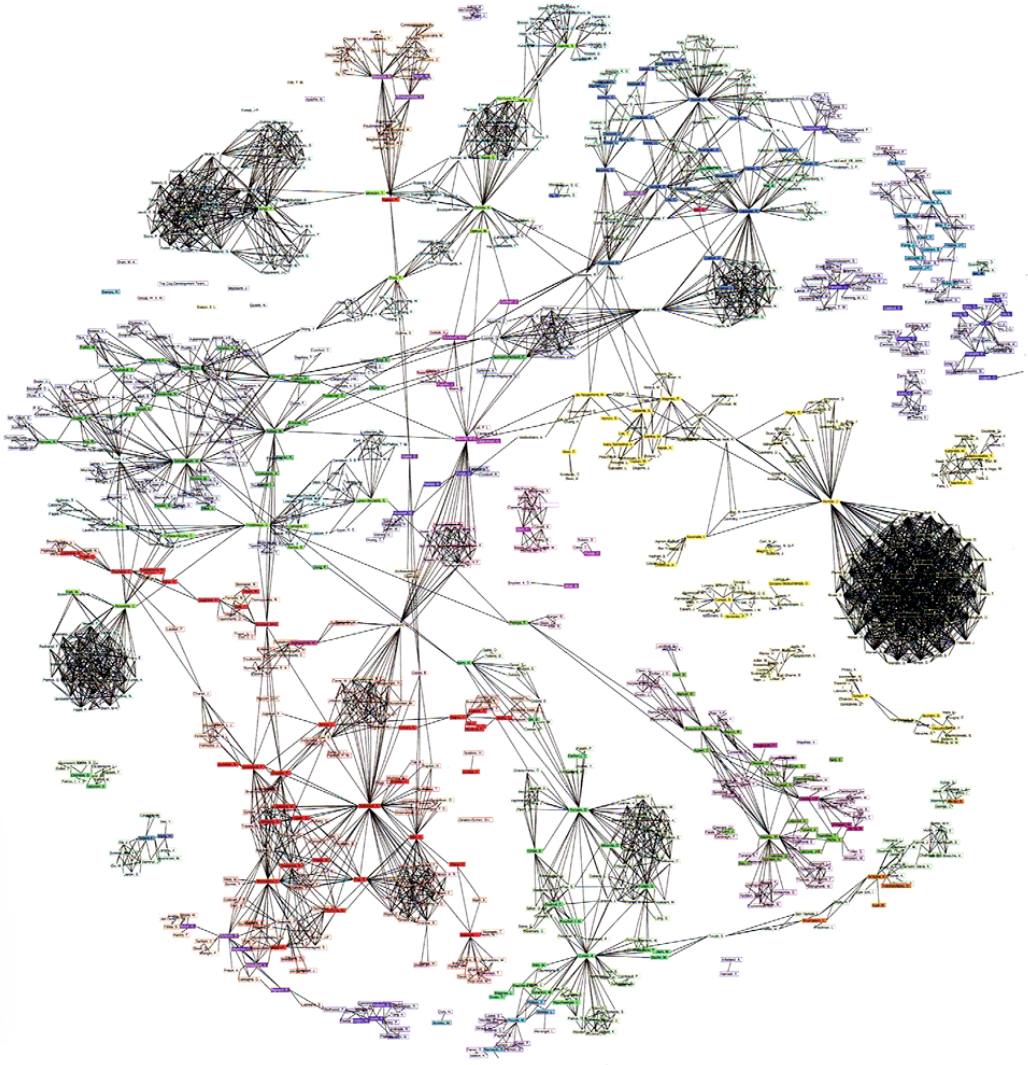


**Fig. 2.29** *LRI Co-authorship Network* (Jean-Daniel Fekete, 2007)
This network represents the coauthorship of the papers written by the members of the Laboratoire de Recherche en Informatique, located in the Univerisity Paris-Sud. Similar papers are placed closer together which creates clusters, effectively dividing the data into visible groups.

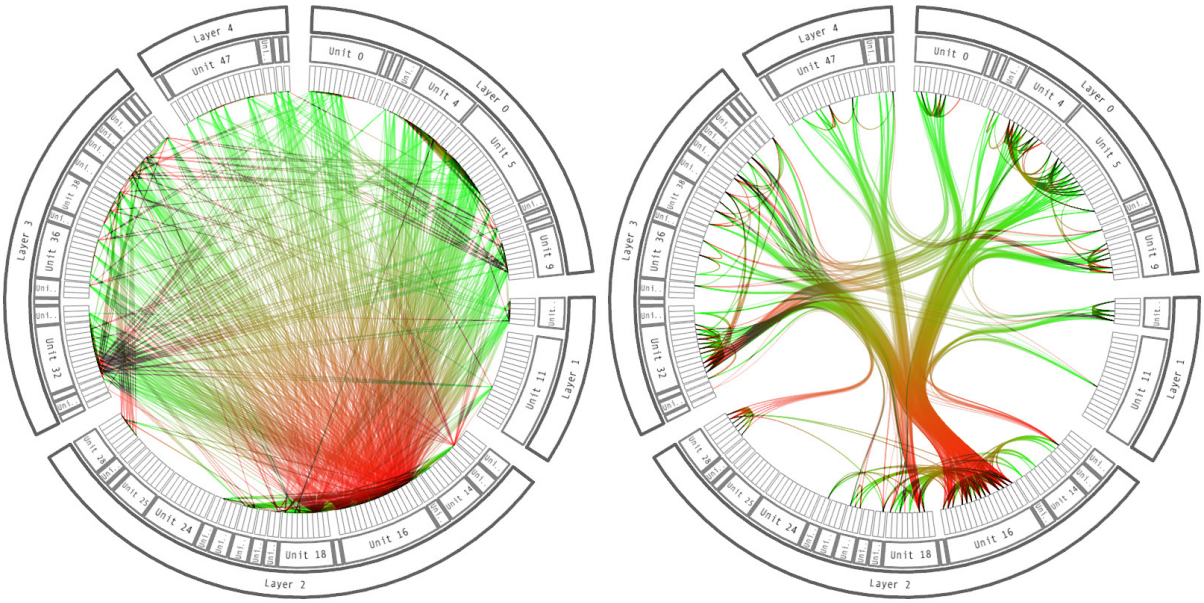



**Fig. 2.30** *Hierarchical Edge Bundles* (Danny Holten, 2006)
This visualization was generated using an edge bundling technique. This specific technique can be used on data sets containing both hierarchical and non-hierarchical data, and bundles the links into more organic structures which are easier to interpret.

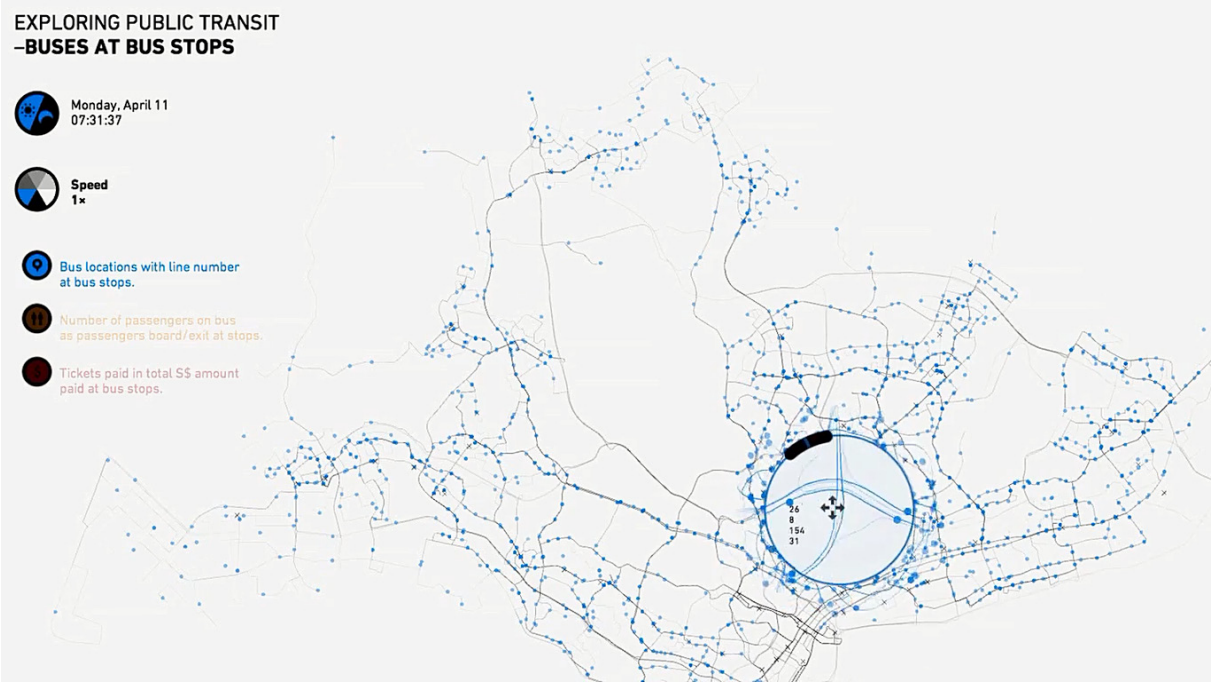



**Fig. 2.31** *Data Lens* (Pedro Cruz, Senseable City Lab, 2012)
This is a map visualization of the bus activity throughout the bus stops in Singapore. The lens are an example of the interactive visualization techniques and algorithms that seek to make high-density visualizations more understandable, as a lot of data can overlap or simply not leave sufficient space for additional information to be readable. Their purpose is similar to that of a magnifying glass, amplifying areas which the user selects and filtering different kinds of information such as the bus number, number of passengers and the total fare payed by the passengers at any one bus stop.

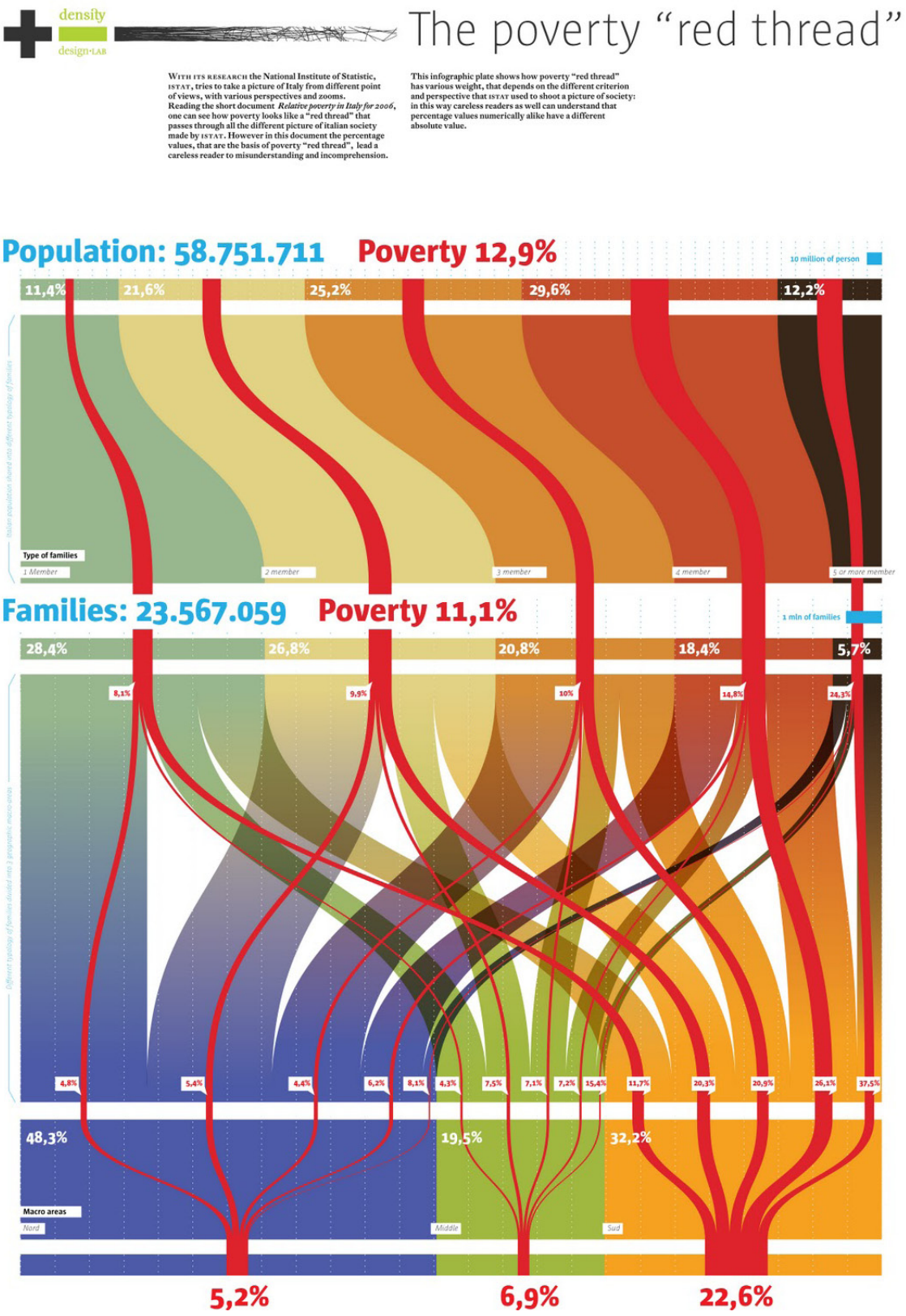



**Fig. 2.32** *The Poverty Red Thread* (DensityDesign, Mario Porpora, 2008)
An example of a flow chart, where the nodes are positioned on opposite sides of the plane, and the links are represented with organic lines. It shows the poverty in Italy, organized the number of members in each family and by their location.

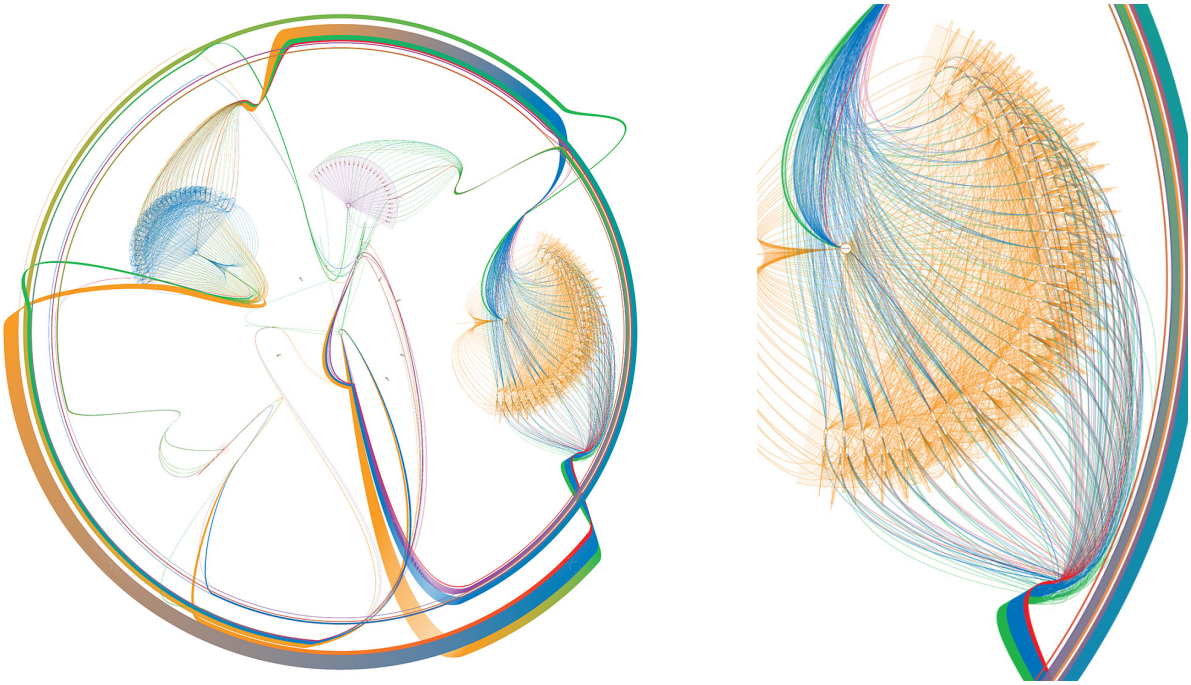


**Fig. 2.33** *Website Traffic Map* (James Spahr, 2003)
This visualization was created using a software tool that illustrates the navigation flow of a website. Instead of mapping simple straight lines between the pages, it represents all the traffic as moving in a clockwise direction.

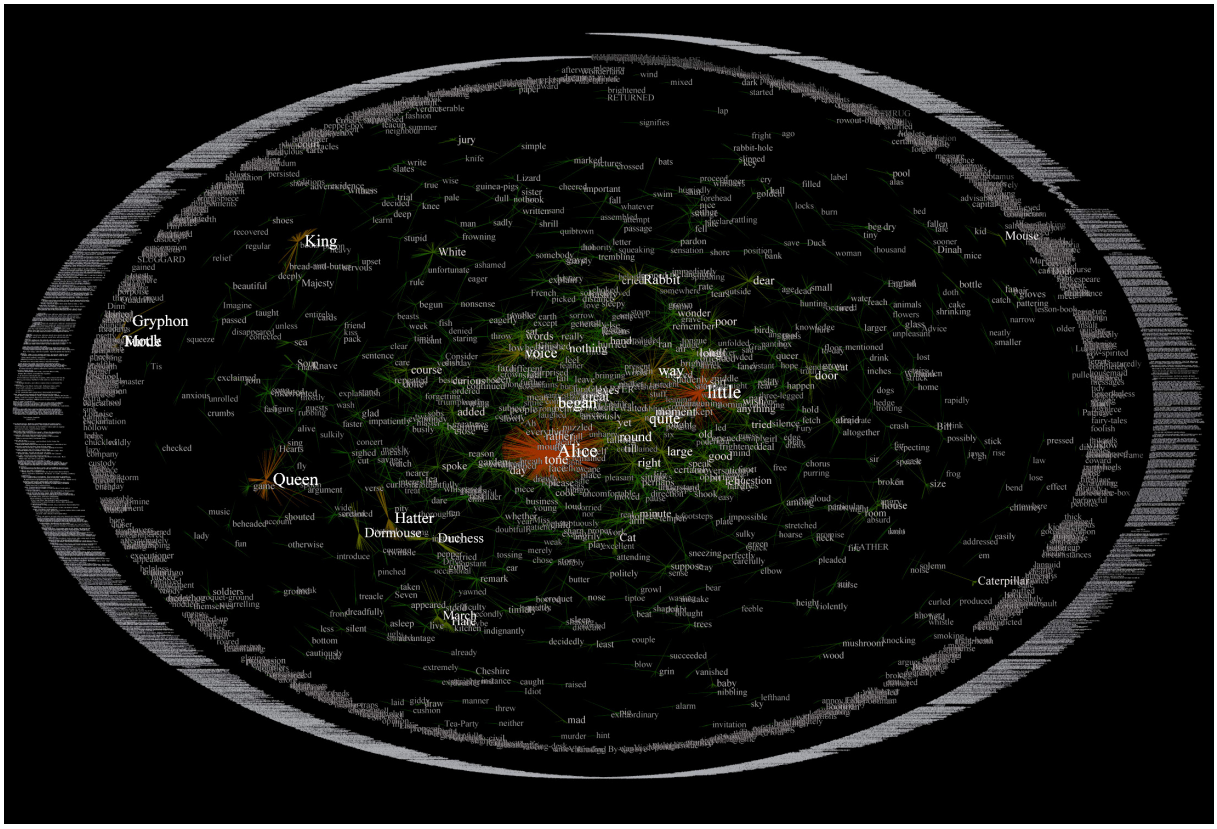



**Fig. 2.34** *TextArc: Alice in Wonderland* (W. Bradford Paley, 2009)
This elliptical implosion network was created by the tool TextArc. The visualization is interactive, highlighting word frequency and associations in *Alice in Wonderland*.

# Hierarchical structures

*Hierarchical structures* are composed of ordered sets of data where elements are organized in relation to each other and to the whole system. They can be divided into two representations, though these may also exist simultaneously: *stacked* and *nested* schemes. In a stacked scheme the elements are arranged in a directional relationship, while the hierarchy is determined by their position in relation to each other. A nested scheme is constituted by elements grouped by containers, positioned according to their interdependency and subordination.

A *tree* is the most basic representation of an hierarchical structure, and it has been present in the history of visualization for over eight hundred years [9]. It is an undirected network with no closed loops where there is only one possible path to travel between any pair of nodes. As its name indicates, this strucutre resembles a tree, where there is a root node which can be connected to multiple other nodes by branches, and these nodes can then branch out to other nodes and so forth (Fig. 2.35). It can be drawn with any node as its *root*, though there is a specific node which is considered to be the origin point. The end-nodes (those without sub-branches) are called *leaves*.As a tree increases in relations, it can spread in all directions throughout the plane creating a *circular tree* (Fig. 2.36), although without proper ordering, the different stages of the tree might not be perceptible.

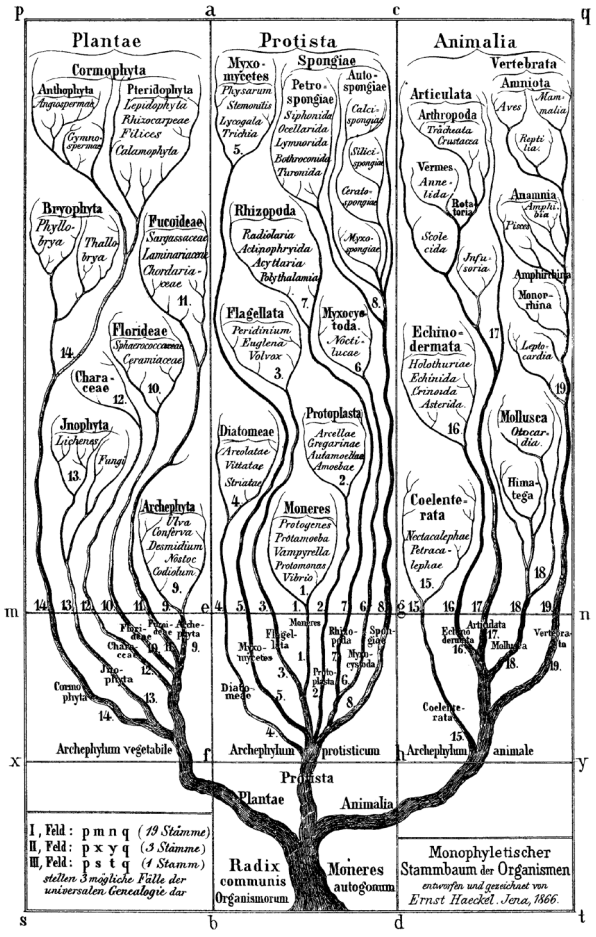



**Fig. 2.35** *Monophyletic Family Tree of Organisms* (Ernst Haeckel, 1866)
An early branching diagram representing three of the kingdoms of life: Plantae, Protista and Animalia.

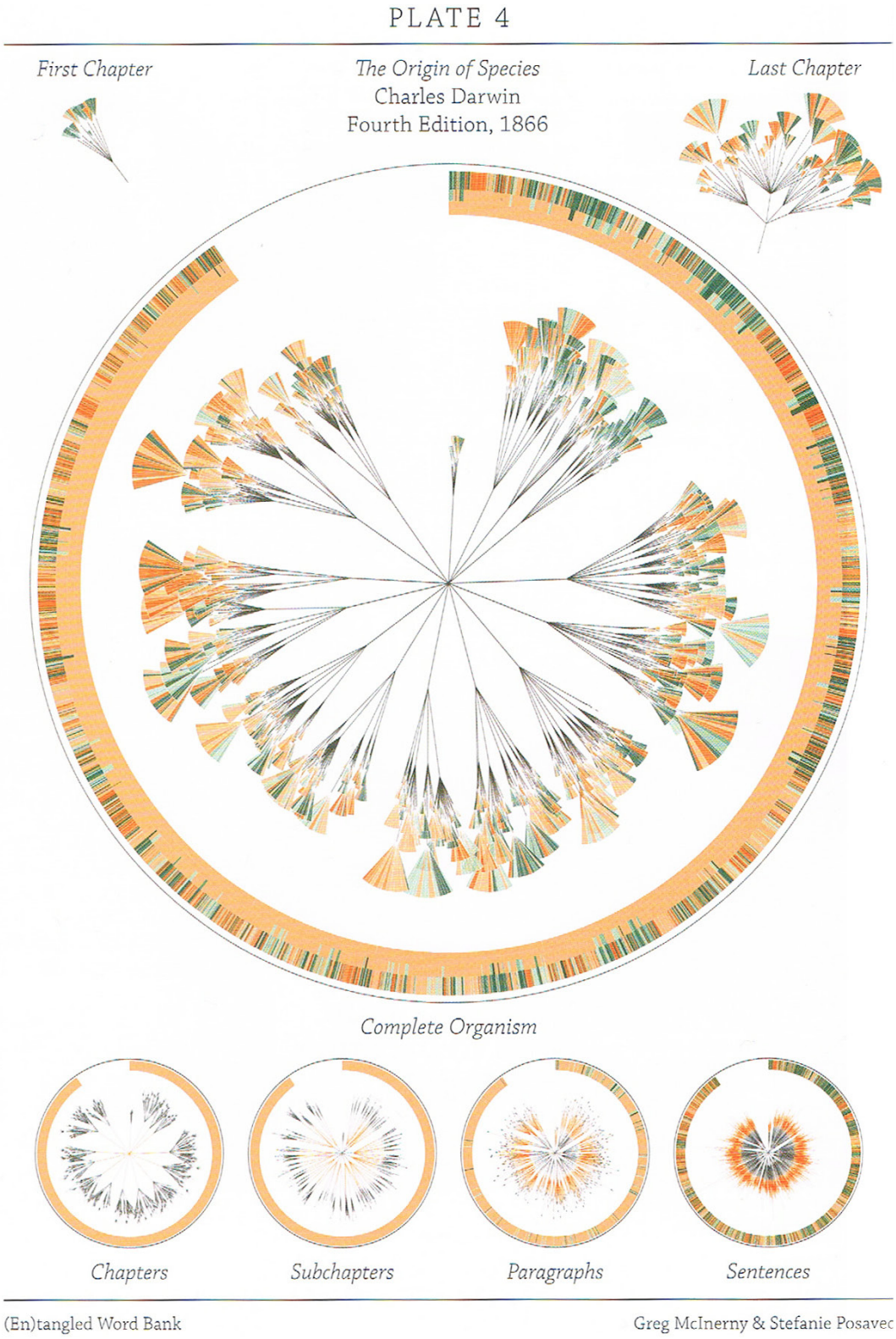



**Fig. 2.36** *(En)tangled Word Bank* (Stefanie Posavec, Greg McInerny, 2009)
This is one of a series of diagrams representing the changes between the six editions of Charles Darwin's *The Origin of Species*. The initial branches represent the chapters, that divide into subchapters, then into paragraphs, and finally into sentences. Blue represents sentences that made it into to the next edition, while orange represents deleted sentences.

The creation of trees can also be a complex process, especially when dealing with large amounts of data, which mainly consists of identifying the position of the nodes. One way to calculate these positions can be done based on what stage each node is on (how many branches away it is from the root node) and how many neighbours they have in order for these and their branches to not overlap each other, as well as maintaining an identifiable hierarchy.

A method that can be used to position the nodes without previous calculations is by simulating forces in order to repel nodes that aren't connected and allow the tree to reorganize itself. The process is an animation which can be more visually interesting than other options, but it may also be difficult to keep the branches from becoming "tangled" on very large trees. It is also possible to utilize a combination of both methods to achieve better results.

Trees can also be represented through more unconventional forms, such as *treemaps*. These represent the nodes and branches using rectangles instead of separate elements joined by lines. Each branch corresponds to a rectangle and its inside is divided into smaller rectangles representing the sub-branches, usually using size to represent the dimensions of the data (Fig. 2.37). Depending on how color and size are handled it might be easy to identify patterns among particularly large datasets.

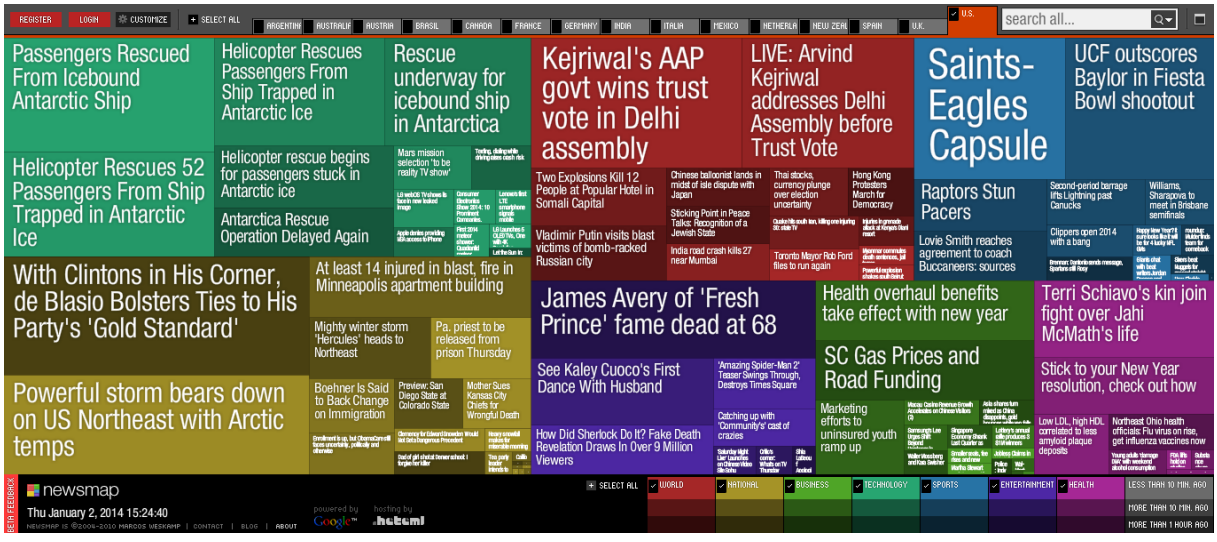



**Fig. 2.37** *Newsmap* (Marcos Weskamp, Dan Albritton, 2004)
A treemap that provides an overview of various online news stories. It is an attempt at visually revealing hidden patterns in the news media by exploring the relationships between the reported news.

# Temporal structures

*Temporal structures* represent other elements in relation to temporal data, such as showing when events occured or the progression of the proprieties of various element during an interval of time.

*Time* is an abstract concept, but we have ways to represent it visually, or at least its passage, using graphical elements and techniques that allow us to interpret objects or values as they change from a temporal starting point to an end point, usually showing the various instances of that element through various intermediate steps.

One of the more common visual proprieties used to represent time is position, usually by mapping out the temporal values on one of the axis of the plane, and as the elements travel across that axis they represent their state for that corresponding point in time (Fig. 2.38). This is known as a *timeline*.

These structures are usually used to represent others with the added dimension of time, like *spatio-temporal structures*, which utilize information with both spatial and temporal elements, and as such, most of the visulizations that fall within the definition of these structures are the ones that show elements on maps changing over time (Fig. 2.39). To properly represent these kinds of changes on a map usually requires approaches such as animations, interactions or multiple iterations.

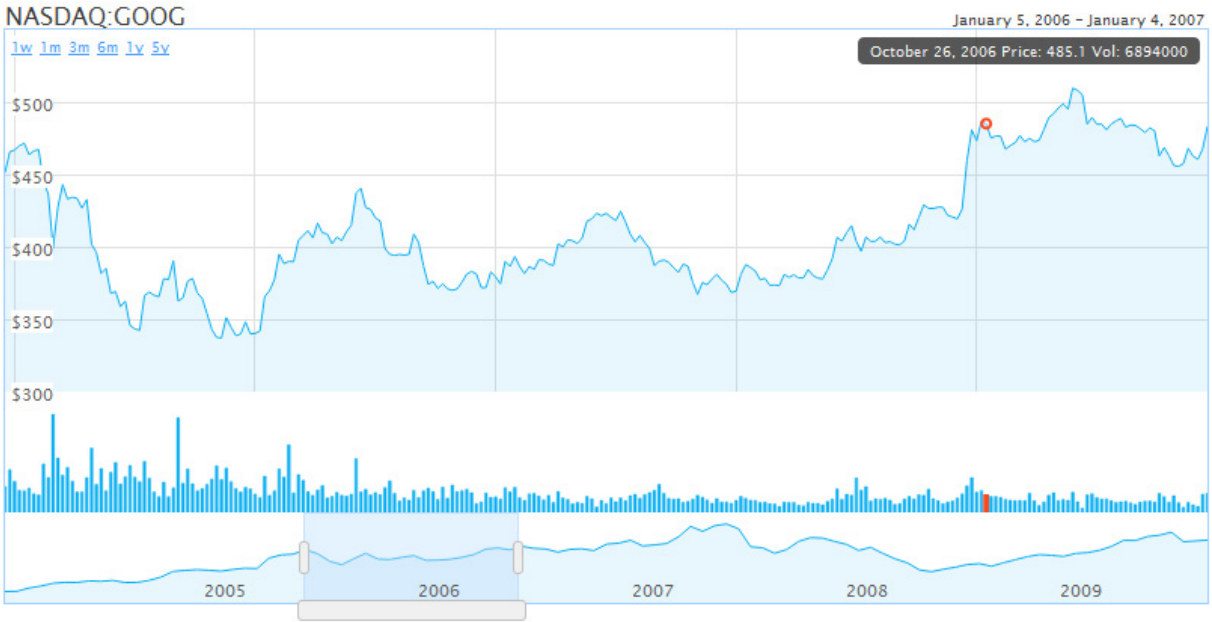



**Fig. 2.38** *Humble Finance* (Humble Software Development, 2010)
A simple interactive HTML5 data visualization tool which can be used to display any two-dimensional numerical datasets and allow the user to compare the two. The image shows it displaying historical stock data gathered from Google. On the bottom there is an interactible timeline which allows the user to pick a range of values to visualize above, as well as the ability to simply scroll through the displayed time.

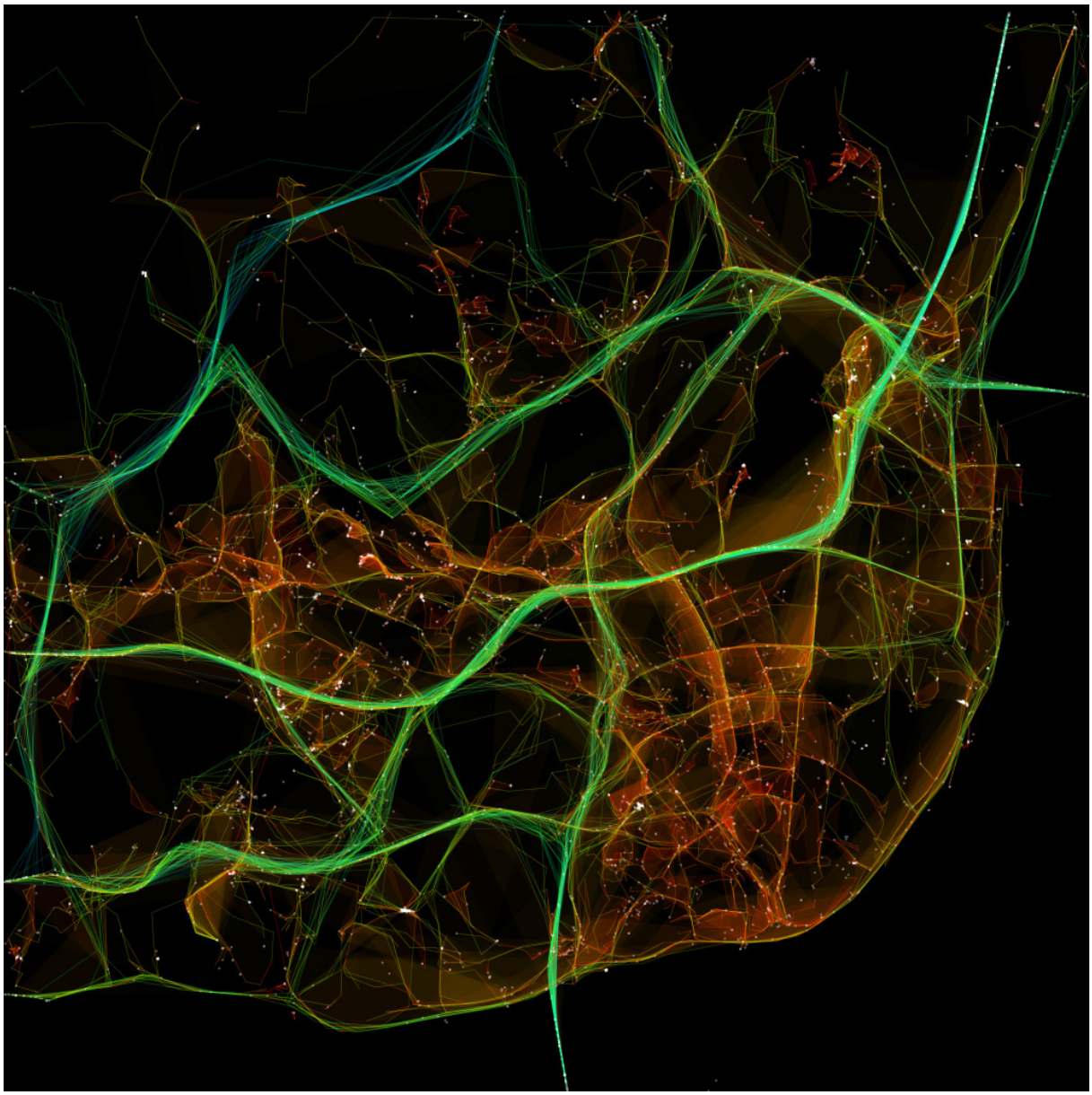


**Fig. 2.39** *Traffic in Lisbon* (Pedro Cruz, Penousal Machado, João Bicker, 2010)
An example of a spatial-temporal structure. This visualization of Lisbon's traffic is animated and shows the traffic changes throughout a twenty-four hour period, which is actually composed of vehicle data gathered during the month of October of 2009. Green and cyan colors represent faster vehicle average speeds, while yellows and red represent the more congested paths filled with slower vehicles.

# Nature-inspired methods for data organization

Representing data visually can be a complex task as we aim to make sense of all the information and induce useful insights through graphical elements, so we develop new methods to create and convey these visualizations. These methods look at how certain living beings interact in nature and convert these behaviors into simple rules which can be applied to the data itself and promote self-organization behaviors. Greg Judelman lists some rules by which organisms in mature ecosystems live and notes: “These properties can be mapped to visualization design in the context of the optimization of the cognitive potential of visualization users“ [10].

As we will be dealing with particularly large sets of data, knowledge of these methods can be useful in order to discover new relationships or organically organize data. It is because of our current technologies that it is possible to represent datasets with hundreds of thousands of individual values and by creating programs that follow simple rules between each one of these values. This way, it is possible to recognize a number of elements on screen and change their behavior based on their current conditions for every frame of the visualization, in other words, it is possible to program behaviors into the elements representing the data so that they react in relation to each other, their data or even other exterior elements. The computer’s processing speed and real-time animations have allowed us to explore a whole new dimension, past the representation of time.

The various elements that will shape the visualization can be programmed with behaviors which can vary in complexity, though even simple rules can generate complex behaviors. One of these elements that has a set of objectives and acts autonomously in an environment which it can recognize is called an *agent*. A system will generally have multiple agents, and when these agents do not have a central coordinator and instead act autonomously, based on other agents in their neighborhood and environment, then this system is called a *decentralized multiagent system*. We will be focusing on these types of systems, as they are inherently capable of self-organization and simulating collective behaviors which we can find in nature, such as in birds or ants. Through these collective behaviors it is possible to observe emergent phenomena, in other words, perceived complex actions that resulted from a group of agents not explicitly programmed to do so.
According to Andrew Vande Moere [11], an average data agent must have certain characteristics:
*Data Interpretation* – An agent must be aware of the data it represents and detect changes made by user interaction or by the application’s timeline.
*Local Perception* – Each agent must be able to perceive their environment, and detect other agents in their neighborhood.
*Local Communication* – An agent should be able to trade information with other agents in its vicinity.
*Negotiation* – The ability to perform more complex trades with their neighboring agents, such as swapping position.

*Visual Presence Autonomy* – An agent's ability to change their own position and visual variables, as well as that of their neighboring agents, to some degree.
*Historical Memory* – The ability to store past values such as previous positions and data from other agents it has encountered.

We will analyze three types of behaviors possible in a self-organizing data visualization model, starting with *particle animation*. In a particle animation, each particle is an agent that follows a set of sequential behavior rules, and moves around in a virtual space using dynamic animation. The particles are subjected to "forces" which attract and repulse them and the rules determine what forces the particles are attracted to and whether they should speed up or slow down. Agents are randomly distributed at the start and move freely, reacting to the forces accordingly and creating emergent visual effects that convey meaningful information over time (Fig. 2.40).

Another behavior is *swarming*, based on the mathematical simulation of flocking birds. Craig Reynolds [12] modeled the movements of what he called *boids* (bird-objects), by creating a set of simple rules that dictate their movement. In this case, the agents are given limited vision and some communication capabilities, and they must follow the Reynold's rules based on rel bird flock behaviors: avoid collisions, match the velocity of agents in the neighborhood, and stay close to the center of the flock. Other rules can eventually be added, such as an attraction to agents with similar data values and an opposite reaction to those with dissimilar data (Fig. 2.41).

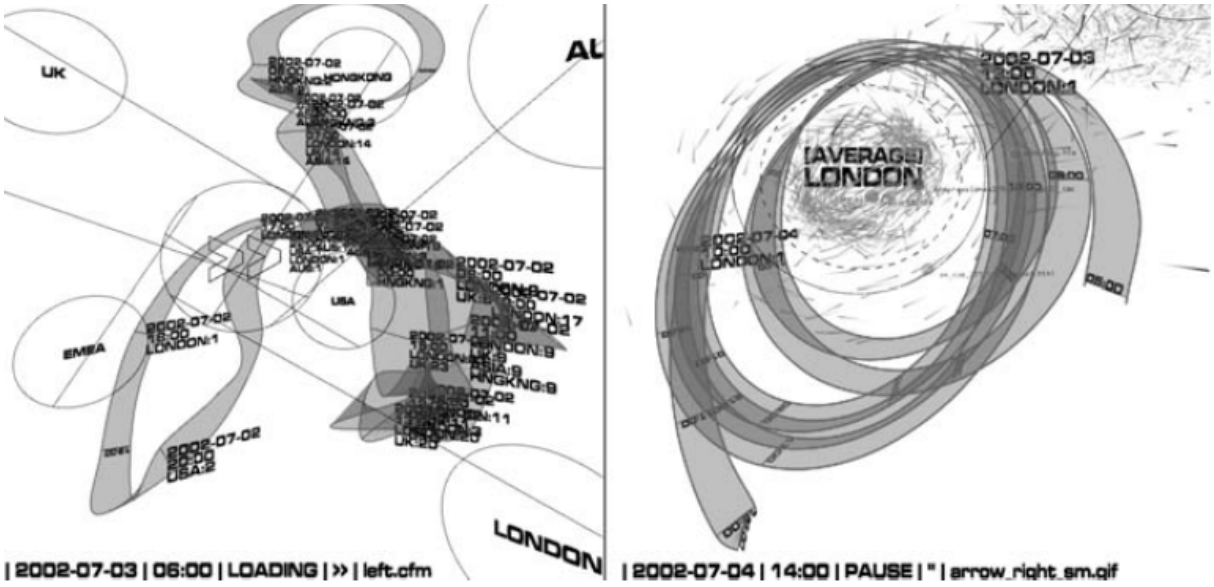



**Fig. 2.40** Particle animation applied to a dataset of the Intranet file usage in a company with about 7000 employees over the course of a year, where each agent represents a document stored on the Intranet file servers. The circles represent the different geographical units, while the ribbon and time stamps trace the three-dimensional trajectory of a particle. On the left, a Quark pattern as emerged, showing documents downloaded often by various different regions, creating erratic movements. The image on the right shows a Comet pattern, representing documents downloaded often by the same region, creating elliptical tracks.

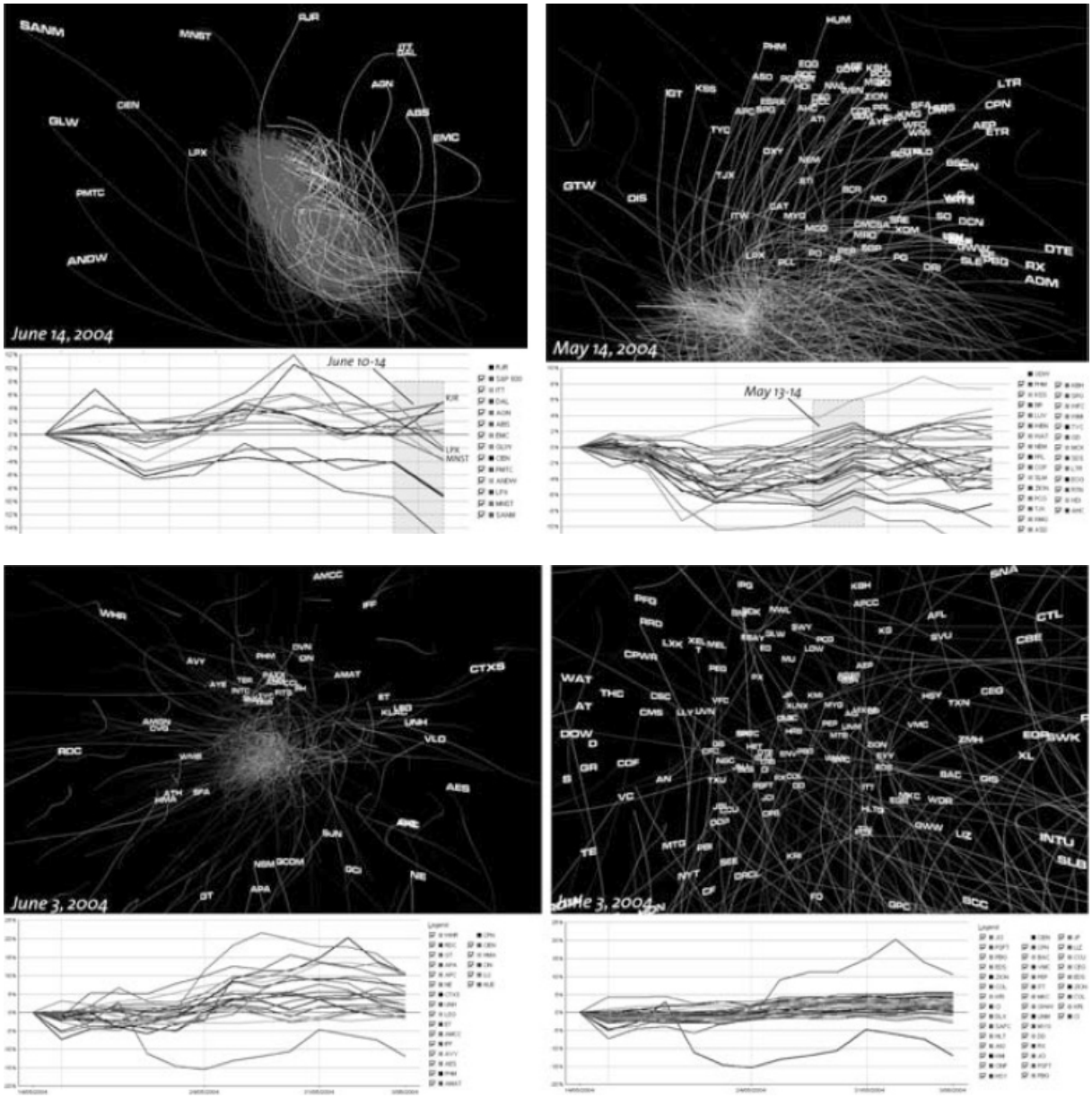



**Fig. 2.41** An application of a swarm of boids showing one year of stock market quotes from around 500 companies. Each agent represents a company, and for each update they calculated and compared the relative difference in price with their previous update, reflecting the change in stock market quotes over time. The top image shows short-term clustering patterns, where individual agents are expulsed by the main flock on the left, and the right image shows subflocks with similar value changes appearing from the main flock. The bottom image show long-term zoning patterns, with the image on the left shows the flock core and the image on the right the flock periphery. The line graphs on the bottom of the images convey the relative volatility of the stock quotes.

The last nature-inspired self-organization method we will overview is the *cellular ant method*, which consists of two approaches: *cellular automata* and *ant foraging*. Cellular automata, originally proposed by Von Neumann, consists of a grid with cells whose actions generally are dependent on the states of neighboring cells (Fig. 2.42). Ant foraging is an example of ant-based data mining, where the agents simulate ants and pick up data items and move around until they find similar data items, at which point there is a probability of dropping it in their vicinity; this originates visual data clusters. However, cellular ants don't pick up data items, they represent the data items themselves and can decide whether to move around or to stop (Fig. 2.43).

Cellular ants follow five behavior rules [11]: *surface tension* (free movement until that agent has enough similar neighbors), *edge repulsion* (moves away from dissimilar neighbors), *positional swapping* (chooses random directions and swaps with neighbors if they meet the conditions), *color determination* (once the collection of ants is sufficiently ordered, colors are introduced where there are stable clusters), and *shape size adaptation* (each agent is able to map one of its data attributes onto its size by negotiating with its local neighbors).

**Fig. 2.42** *Game of Life* (John Conway, 1970)
An example of a cellular automaton where cells have discrete states which can switch between being "alive" and "dead" based on the state of their surrounding cells. A simple set of rules creates emergent visually intriguing cell patterns. The image shows a pattern called Glider Gun, discovered by Bill Gosper, in which the top cells repeatedly move from left to right and generate patterns of cells that move diagonally to the bottom right.

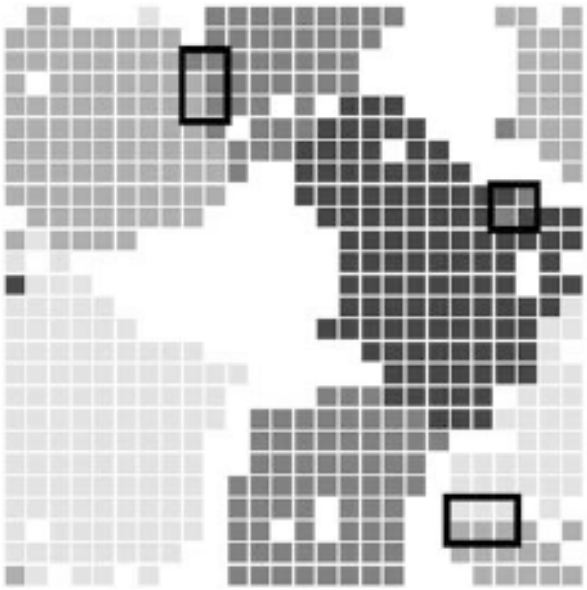


**Fig. 2.43** This image shows a simple visualization of five-hundred ants distributed across four classes, utilizing a two-dimensional synthetic dataset. Initially the ants are placed at random, but through the rules of cellular ants described previously they form clusters according to data similarity. Aside from the perceivable clusters of similar data, we can observe other emergent phenomena such as agents on the border of two clusters that also share similar parameters, highlighted by the black border areas.

These nature-inspired techniques have been used mostly for generative art, creating fluid forms and organic flows [13]; however, as Andrew Vande Moere [11] has demonstrated, there is great potential in applying these data-organization methods to the visualization of information. Not only do these methods contribute to the emergence of visual patterns that can convey meaningful information and induced useful insights, but the very nature of the animation that leads to these patterns serves as a way to develop interest in the viewer (Fig. 2.44), as well as functioning as a generally more efficient way to convey the information, as the viewer can observe the organization process as it happens and the patterns form.

**Fig. 2.44** *An ecosystem of corporate politicians* (Pedro Cruz, 2013)
This is a more recent example of an interactive visualization that employs these techniques, showing the relationships between the members of the Portuguese government and companies and other groups from 1975 to 2013. Companies are represented as circles and their size is proportional to the amount of politicians that held a position in that company, while the politicians are represented as small creatures whose color is based on their most recent political party affiliation and with a body shape that changes based on the number of companies is has held a position in. The data serves as an ecosystem and each politician has a sequence of companies to visit, and as the politician agents enter the visualization through the top, they move around randomly through the companies, searching for companies in which they have held a position in, at which point they will circle around it and search for the next one.

# Brief history of genetic algoritms

Evolution describes the process by which different species and living organisms have changed throughout time, usually involving their gradual adaptation to a new environment or changes to their current one. It has long been the target of discussion in many fields, from science to religion and even philosophy, as humankind desires to explore and determine not only theirs but the history of that which surrounds them.

Charles Darwin's *The Origin of Species* [14] remains at the core of this concept as one of the first theories to be presented with a firm foundation in reasearch and evidence, and it was eventually embraced by the scientific community. Darwin's theory describes natural selection, a process where populations keep favorable traits while discarding undesired characteristics through future generations, which comes as the result of competition between or within species where the most apt individuals have a higher survival rate, and thus are more likely to reproduce and pass on their genes which cointain these advantageous traits for survival. While natural selection explains how species adapt to their surroundings, Darwin also had another theory regarding how the traits relating to the competition for the oposite sex evolve called sexual selection, which wasn't as widely accepted. This theory refers specifically to the characteristics pertaining to the struggle to reproduce, where males need to eliminate their competition by either driving them away or killing them, while females tend to try and attract or excite agreeable partners [15]. This becomes quite apparent in birds, such as the peacock, where the vibrant plumage is used to attract mates and those with unfavorable colors and designs do not get to pass on their genes.

The idea and subsequent studies of creating tools based on evolution which could be used for optimizing solutions for engineering problems started around the middle of the twentieth century. The concept was to evolve a population of candidate solutions by simulating genetic variations and natural selection. During this time, a few evolution-inspired algorithms were developed by researchers such as G.E.P. Box (1957), G.J. Friedman (1959), W.W Bledsoe (1961), and H.J. Bremermann (1962), but it their work did not attract much attention. In 1965, a more successful technique was introduced by the German researcher Ingo Rechenberg which he called "evolution strategies". These were used by Rechenberg to optimize solutions for aerodynamic wing design and their success lead to future developments which kept these strategies in use even in the present.

One of the more important actors in the development and popularization of genetic algorithms was John Holland, who had been studying the adaptation process in nature and how it could be used in programming, along with his students and colleges at the University of Michigan in the 1960s and 1970s. Holland published a book in 1975 entitled *Adaptation in Natural and Artificial Systems* [16] which contained a lot of his work presented in a detailed and systematic way in order to further establish the presence and

possibilities of genetic algorithms, as well as present the newer concepts of simulating evolution and natural selection processes as a tool for solving problems through adaptive computer programs using selection, crossover and mutations. His work proved to be an innovation and he was the first to attempt to put computational evolution on a firm theoretical footing, which had served as the basis for almost all the subsequent work in the field of genetic algorithms [17]. By the early 1980's, researchers had already become more interested in genetic algorithms and it's applications exponentially grew alongside the technological upgrades and development of newer tools. Genetic algorithms were applied to a multitude of fields of study outside of pure mathematics as they can be used to solve various complex problems much faster than the human mind.

Genetic algorithms have been applied in several real life contexts. In 1994, Victor Johnston started development on a software that used genetic algorithms to evolve human faces through the input of a user which he called *FacePrints* [18]. This software was used as a study for human facial beauty by surveying users who would look at random faces displayed and rate them, and the software would close in on their perception of the most beautiful face. It was also put to a more practical use in law enforcement, being used as a substitute for sketch artists, helping witnesses describe and identify suspects (Fig. 2.45). FacePrints would combine a variety of faces in different ways and would adapt according to the responses, making it easier for most people which had a better time identifying whole faces rather than describing the details.

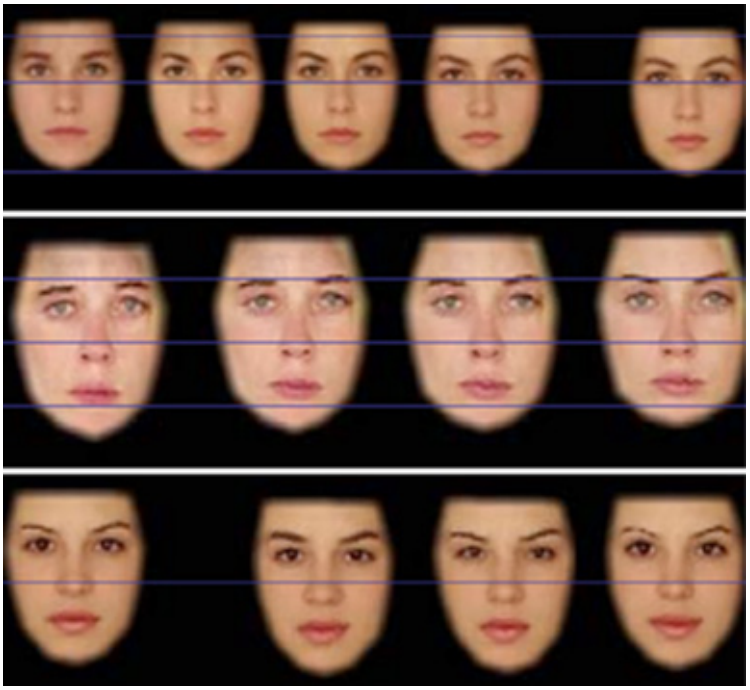


**Fig. 2.45** Preliminary results of Johnston's FacePrints software showing how features are switched and altered in the attractiveness evolution of a face.

In the field of electrical engineering, Edward Altshuler and Derek Linden (1997) created a genetic algorithm that would determine the design of a wire antenna based on certain pre-determined proprieties, which up to that point used to be a process that relied on experience and intuitive guesses [19] (Fig. 2.46). The application of genetic algorithms even extended to aerodynamics, where Shigeru Obayashi, Daisuke Sasaki, Yukihiro Takeguchi, and Naoki Hirose developed an algorithm which would determine the shape and proprieties of a supersonic aircraft's wings, taking into account the advantages and disadvantages of each propriety and letting the program resolve which were the most overall advantageous tradeoffs [20] (Fig. 2.47). While these examples show some real life applications of genetic algorithms, the field of evolutionary computation is still developing new approaches, including studies now focusing on the theory of sexual selection and how it could help produce new results through genetic programming [21] [22].

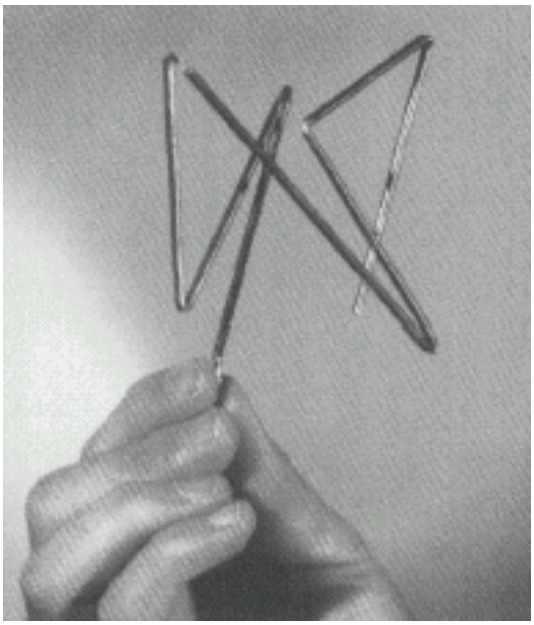


**Fig. 2.46** An example of a wire antenna and it's peculiar design which would have to fit the predetermined parameters in order to function correctly. The genetic algortimn could reach the most efficient designs much quicker than a human using trial and error.

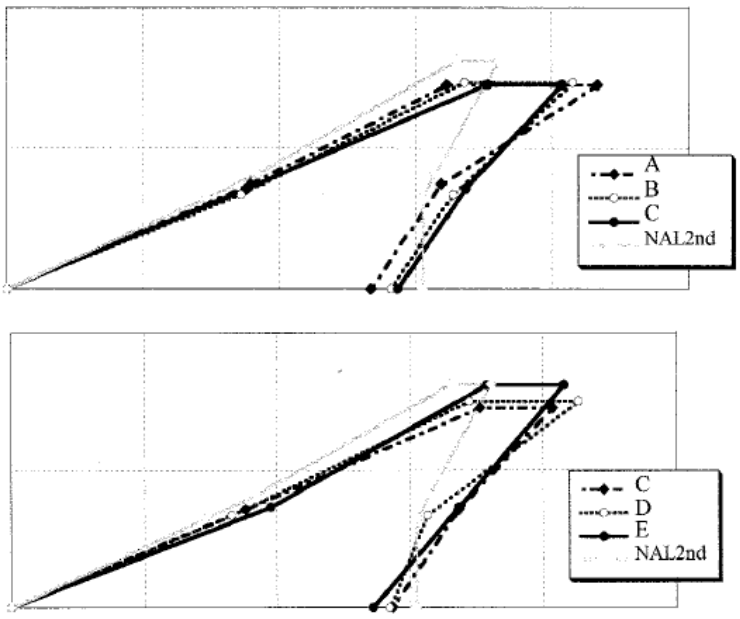
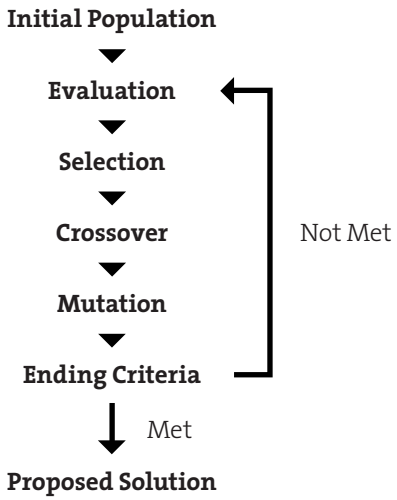



**Fig. 2.47 Some examples of the** variations of the aircraft's wing design which were created by the variation and manipulation of six variables.

# Genetic algorithms

Genetic algorithms simulate the process of natural selection in order to find a solution to a problem, usually relating to optimization and searches [23]. The problems solved by these algoritms may not have exact solutions, so they search for one which is as close as possible through recombinations. Random solutions are generated and then progressively evaluated and recombined to reach better results, until a certain criteria is met, at which point the proposed solution will be the one with the highest evalution up to that point.




In order to explain in more detail we will use a simple problem that can be solved through a genetic algorithm as an example. This problem consists of finding a path between two points where a ball released on the starting point would reach the end point in the least amount of time (Fig. 2.48).

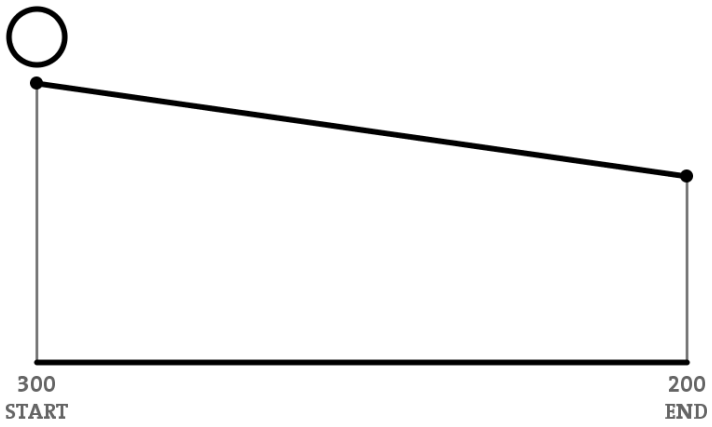



**Fig. 2.48** A path is drawn between the fixed starting and ending points which the ball will travel on.

For this particular problem, the path will consist of eight points equally spaced apart on the horizontal axis, so each point only varies in their vertical position, reducing the sought-after solution to a set of eight numbers (Fig. 2.49).

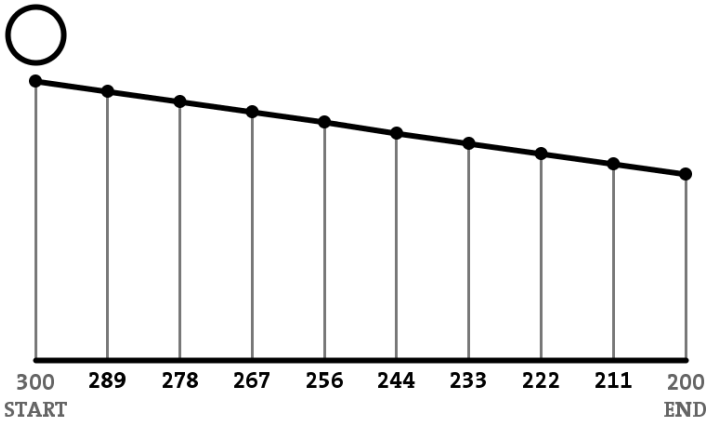



**Fig. 2.49** This illustrates a possible solution to the problem, depicting a curve consisting of a start point and an ending point with eight intermidiate points in between, evenly spaced out on the horizontal axis and with varying heights.

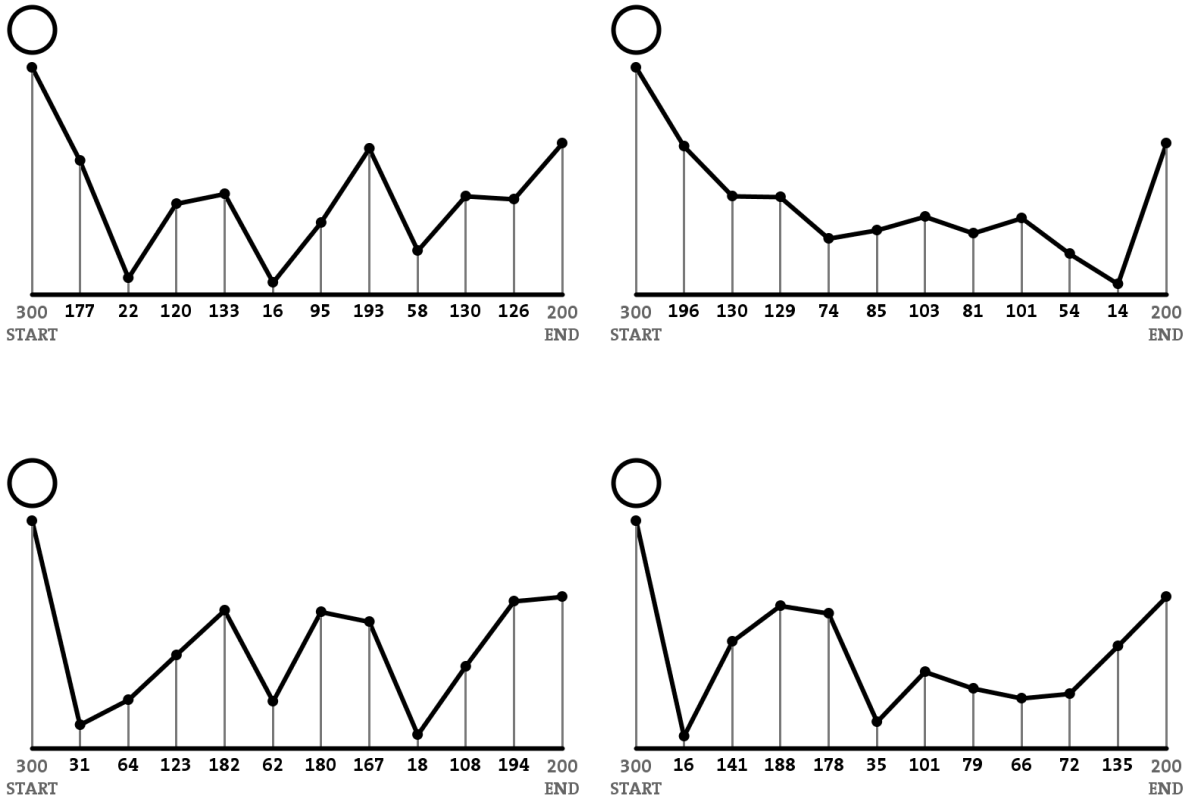



**Fig. 2.50** This group of graphs represents a part of a first generation of individuals, which were randomly generated. Their genotype is depicted under each graph: a set of numbers that correspond to the height of each point that makes up the path.

The algorithm is initialized by randomly generating a group of possible solutions, and each one of these is called an *individual* (or *phenotype*), while the group is referred to as a *population*. Each individual is characterized by a set of proprieties, its *genotype*, which in the context of this problem would consist of an array containing eight numbers, referring to the heights of each point in the path. This is known as the *first generation* and it is randomly generated (Fig. 2.50).

The next step consists of an evaluation of each individual as a possible solution, which is assigned as a numerical value known as *fitness*. There can be multiple ways to evaluate an individual but these will always be dependent on the problem trying to be solved and its goals. The evaluation of each proposed solution is done through a *fitness function*. In the case of the problem described so far, the fitness function is actually aware of the perfect curve between the two points, as there exists an actual mathematical solution for this, and the fitness of an individual is calculated by comparing it to the target curve (Fig. 2.51).

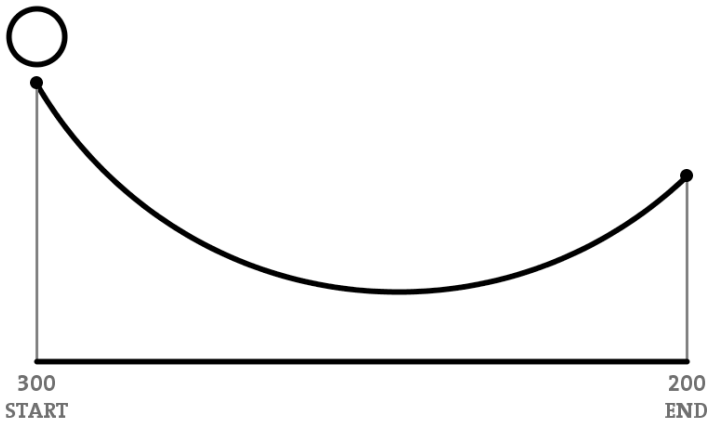



**Fig. 2.51** This image represents the target curve which each individuals curves would be compared to in order to determine their fitness.

After the evaluation, the population is then subjected to a process of selection where a group of individuals will be chosen to breed the next generation. This can be achieved through a variety of methods, some of the more well-known being *tournament selection*, *truncation selection*, and *fitness proportionate selection*. Selection isn't restricted to one method, which means multiple can be used at once. Tournament selection consists of holding several tournaments, starting with randomly selecting a quantity of individuals from the population and then choosing the one with the highest fitness. Truncation selection does not use probability and simply orders the population by fitness and chooses a set percentage of the best individuals. Fitness proportionate selection, also known as roulette-wheel selection, assigns a winning probability to each individual proportional to their fitness, meaning that the best individuals have a higher chance of winning but will

not always be selected, which can add variation to future populations and possibly lead to fitter solutions down the line. There is also *elitism*, which consists of copying the single best individual into the next generation.

The new generation will be created from the selected individuals which will be subjected to modifications caused by *genetic operators*, which involve transformations and recombinations of the genotypes to create new individuals. There are two genetic operators: *mutations* and *crossovers*. Mutation alters one or more gene values of an individual's genotype, replacing it with a different value which is usually random but within established limits (Fig. 2.52). This helps maintain genetic diversity which can prove indispensable in finding better solutions. Crossover involves the creation of a new individual through the recombination of others, usually involving two. A *one-point crossover* is one of the most simple types of recombination, in which two individual's genotypes are split at a *crossover point* and then two of the resulting sections are switched, resulting in two children whose genotypes contain gene values from both parents (Fig. 2.53). In a *two-point crossover* there are two crossover points and the resulting middle section is switched two create the children. Whether or not to maintain the genotype's length will depend on the problem trying to be solved, as crossovers can swap two differently sized groups of genes which happens when the crossover points do not match on both genotypes, this being more commonly known as the cut and splice approach. There are also other types of crossovers which can consist of switching various random gene values between both individudals to create new ones, instead of switching whole segments determined by crossover points. The end result of this stage is a new generation of new individuals which has mostly resulted from previous generations' best individuals.

The previous steps will be repeated in a cycle, where each successive generation will be subjected to the processes of evaluation, selection and recombinations, in order to create better individuals until a solution that is determined to be fit enough is found, or until a specific number of generations has been created. This solution is usually the most apt individual of the final generation, and if the algorithm was successful, then this inidividual should quality as the answer to the problem which the user was trying to solve.

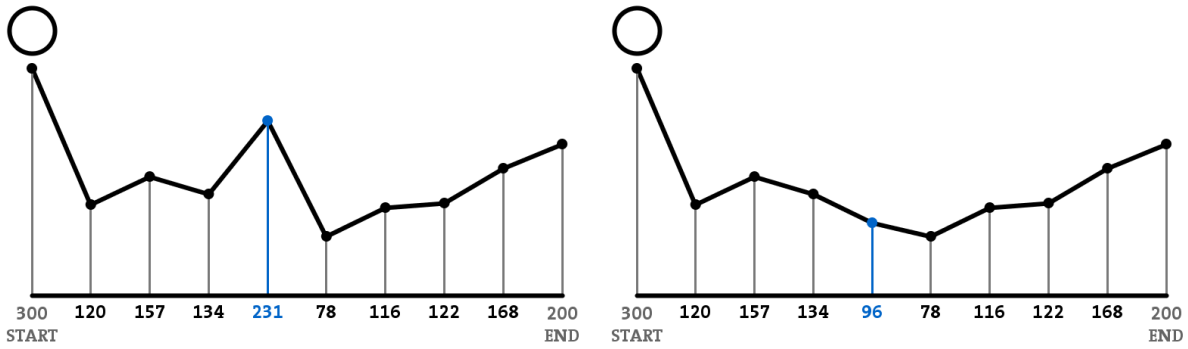



**Fig. 2.52** A representation of a mutation affecting a single gene value in an individual genotype, represented in blue. The mutation randomized that value in order to introduce some variation which could not be achieved with only recombinations.

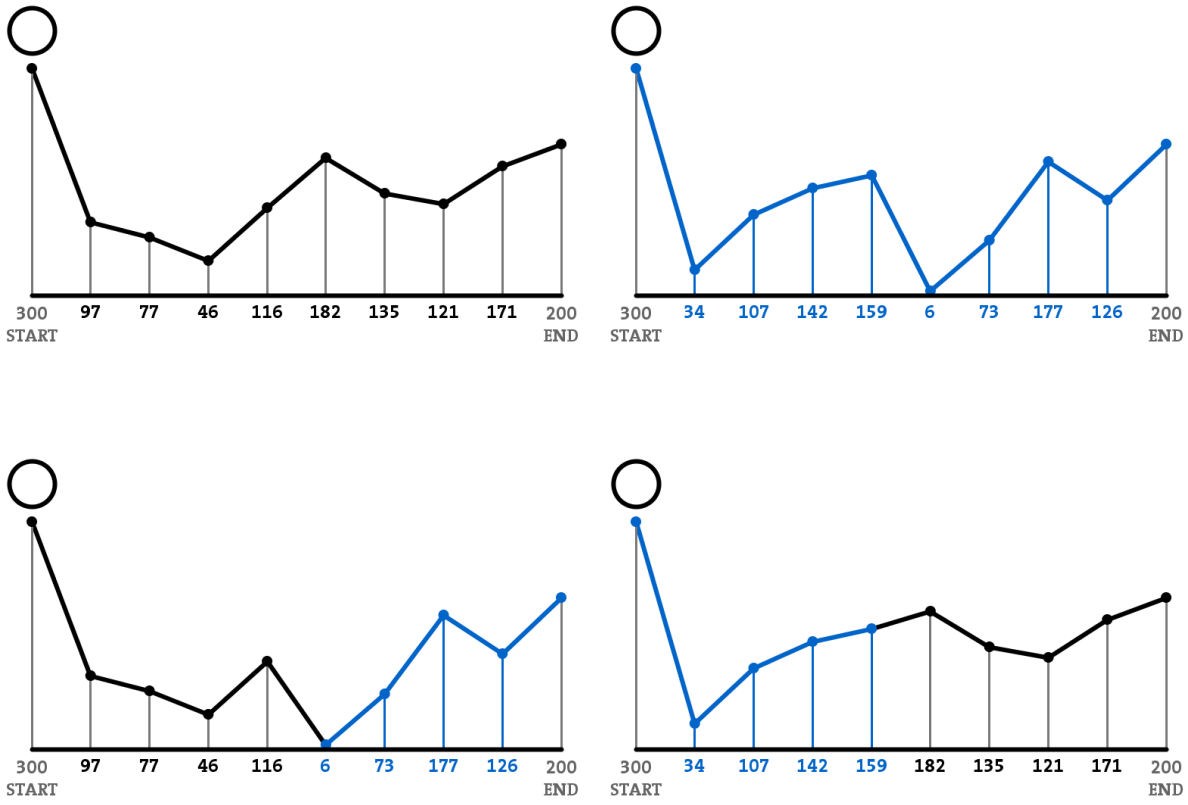



**Fig.2.53** A one-point crossover between two individuals shown in the top two pictures, with a central crossover point which splits them in two equal parts. These two sections are switched, resulting in the two new individuals shown in the bottom two images.

# Chapter 3
# Objectives and methodologies

## Research objectives

Our main objective is to build a functional prototype capable of building, presenting and browsing visualizations capable of enabling the extraction of high-level knowledge. The prototype was developed in *Processing* (an open source programming language based on Java) and will be able to load, read and process external data files and then represent this information on the screen. The application is interactive, allowing for the user to control the information which is being displayed and manipulate the graphical elements to a certain degree, which are essential to maintaining the visualization easy to interpret when displaying very large quantities of data. These interactions include some basic functions such as viewing different levels of detail, which can mean switching between a simple view and one with more textual information on screen or simply the ability to control the zoom level, allowing the user to navigate the data, and adding switchable filters to the data to pick out the most relevant information for each scenario. The implementation of these is largely dependent on their usefulness regarding the choices made in the development process and they can be subjected to changes. While our focus is on producing a dynamic visualization which the user can interact with and visualize animated adaptations or transitions between the data, we also to implemented the ability to produce static data artifacts, though these are subjected to certain limitations such as not being able to show all of the information present in the dataset at once. However, these limitations can be controlled by the user, allowing him to decide the type of artifact to generate.

By focusing on a single type of data from a specific field of study, the visualization can be developed and better prepared as a tool to aid users in this field. The data which we are working with is a set of files contain information about various results from a genetic algorithm, all of which are different due to the stochastic properties of genetic algorithms, as well as some of the results having been created with different settings altogether.

The objective of a data visualization is to assure that the data is represented in such a way that it can convey the meaningful patterns which can lead the viewer to draw useful insights and conclusions. Through some of the theoretical reasonings of Jacques Bertin and Edward Tufte which we examined in the state of the art, we established a set of simple rules which

served as a starting base for building these representations. It was necessary to represent various types of structures in order to represent different types of data with different relationships, including networks and trees. We also used nature-inspired techniques in order to achieve these emergent patterns of information, as they can be used to dynamically organize the data and achieve more visually organic results than through standard functions. The visualization shows off the relationships between the data and provides a clear picture which allows the viewer, or in this case the user, to understand the process that lead to each solution generated by the algorithm.

# Methodologies

For the creation of our project we followed Ben Fry's methodology proposed in *Visualizing Data* [24] as a model, which helped us establish a plan of action for the development process. This methodology consists of seven stages:
*Acquire* – Obtaining the data that will be represented, which involves resolving issues regarding the location and accessibility of the data.
*Parse* – Parsing the data is the process that organizes the data into defined categories that are appropriate and structured to fit into the project requirements.
*Filter* – A selection of the pertinent data that excludes undesired information that does not need to be processed or displayed.
*Mine* – Involves discerning meaningful patterns of information through statistical methods or data mining.
*Represent* – The application of a basic graphical model that presents the information visually.
*Refine* – Improving the basic representation model by making it more comprehensible, visually engaging and aesthetically pleasing.
*Interact* – The addition of elements to the visualization that permit the user to explore the data or even manipulate it, by giving him control over what he wants to visualize and the level of detail.

These steps establish a path from the collection of the raw data to the graphical representation and additional features that make it more visually engaging and make the inference of useful insights easier. However, this is not a strict model, as it admits for the possibility that some steps might be excluded depending on their pertinence to the project. While these steps are presented in a certain order, the actual process can be much more iterative, repeating previous steps in order to obtain progressively better results. Later stages might lead to new interactions with earlier ones because of how some steps affect the others. Fry illustrates this through a graphic (Fig. 3.1).

Our process also started by going through the first three steps which involved obtaining the data and analysing it in order to make some initial conclusions regarding what variables would be represented and which structures would be the most adequate. We created some basic graphic representations of the

data which were an essential step in determining whether or not the results are meeting our expectations, and whether the data needs to be processed differently. After the initial representations, we focused mainly on the last three steps.

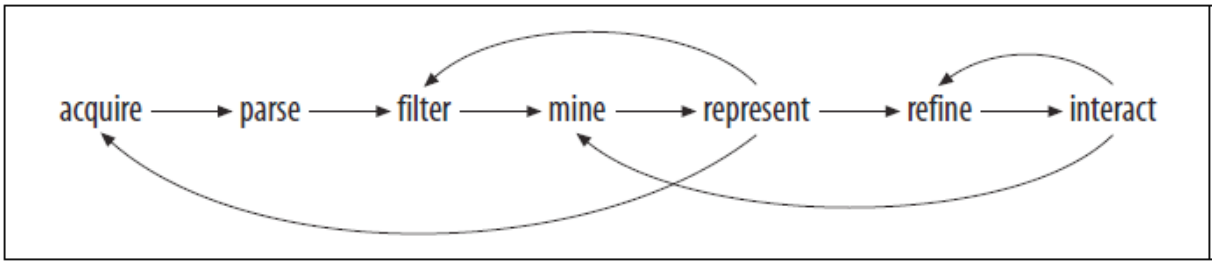



**Fig. 3.1** Small diagram by Ben Fry illustrating a possible iteration chain between steps.

As supported by Fry's methodology, developing both the representations and the interaction is mostly an iterative process based around successive refinements and validations of both the visualization and the interactive elements. In order to represent the data effectively and efficiently, we followed the concepts proposed by Jacques Bertin [1] and Edward Tufte's [2], as discussed in the state of the art, which allowed us to build a strong initial infrastructure for our visualizations and ensure a higher standard of efficiency in communication. Regarding the interaction stage, it was important that the user did not feel hindered by implemented function, so we explored the concept of fluid interaction [25], which is a set of general principles which can be applied to an interactive application in order to support the user's immersion and involvement. The word "fluid" in this case refers to "continuous" or "smooth", since the objective of these principles is for the user's experience to not be interrupted, which can likely happen if, for example, they are unsure how to preform a certain action or if they do not know what a certain action does. A few examples of these principles are: keeping a minimalist interface, smooth and animated transitions, immediate visual feedback, low resistance to change, integrating interface components into the visual representation, and a focus on the experience.

# Chapter 4
# Preliminary work

During the first semester we participated in a project created for a competition hosted by the *Massachusetts Institute of Technology* (MIT), more specifically, the *MIT Big Data Initiative* at *CSAIL* working in partnership with the City of Boston. The competition was named the *Big Data Challenge*, launching on November 12th of 2013 and it was concluded on January 20th of the following year. The Big Data Challenge's proposal was divided into two separate projects: the development of prediction algorithms and the creation of information visualizations. Both of these used a provided group of datasets that contained information about events, points of interest, taxi pick-ups and drop-offs, twitter messages and the weather, all from the city of Boston.

Our initial intention was to utilize the data to create network visualizations that would directly relevant to this dissertation's objectives; however, the data was heavily geographical and a spatial-temporal approach was decided to be much more relevant towards the objectives of the Big Data Challenge. The project required a similar methodology to this dissertation's project for its development, where a large set of data had to be analyzed and filtered and then properly represented through the creation of an interactive visualization. It provided a learning experience through some of the tools and representation techniques used, as well as some of the dynamic, interactive approaches which were later applied.

We created a visualization of the taxi pick-ups and drop-offs in relation to the events in their area in order to show the correlation between the intervals of time and places where events are happening and the people riding taxis to those areas. There were two teams created, one for the data mining portion – comprised of Filipe Rodrigues, Francisco Antunes and Francisco Pereira – and another for the creation of the visualization – composed of Evgheni Polisciuc, Carlos Bento, António Cruz and Penousal Machado. The remainder of this section will focus mainly on the visualization developed by both António Cruz and Evgheni Poliscius with the help of Penousal Machado. The project was developed in Processing, and our objective was to display the events and taxi pick-ups and drop-offs geographically on a map of Boston, and then to make the taxi activities react to the nearby presence of events. We focused on an interval of time where the available data from both taxi activities and events would coincide and also exist in a significant amount to generate interesting and significant visualizations. The interval of time chosen was the month of May 2012. The concept at the base of the project was that each taxi pick-up and drop-off represented a person which would react to events in their vicinity: taxi pick-ups would be repelled by events, representing people

leaving the event by taxi, and taxi drop-offs would be attracted to them, representing people leaving the taxi to attend the event. Graphically, these functions would lead to the creation of animated waves which would give an idea of the movement and flow of the areas where events were taking place. Each taxi pick-up and drop-off was represented as a point which left a trail (a line) behind as it moved, which mapped its course. The color of pick-ups is blue while drop-offs are red, although the lines representing their trails used a gradient color that changes the hue slightly through their length – pick-ups change from blue to green and drop-offs from red to orange – and their opacity and stroke weight changes based on the standard deviation calculated.

Events have two representations in order to distinguish those that have data pertaining to their duration, represented with circles, and those that do not, represented with crosses. Events that do not have an established duration are given a default duration one day, the day in which they take place. The animation of the taxis' activities in relation to the events was achieved using a force-based layout [26] where the force applied is based on their distance to their initial position, which slows each point down as they travel further away until they stop, and on the standard deviation, which determines how likely they were to travel to that event, determining the distance they would travel. Drop-offs are simply attracted to the closest event, though pick-ups need to be repelled by all events and thus required an average of the events nearby in order to move away from them, so their path will not necessarily be linear (Fig. 4.1).

Due to the size of the data we could not have examined every single data entry and have built a visualization that handled each event and taxi activity through a personalized method, so we took a nature-inspired approach where we defined rules and procedures for how each type of data would be represented and interact with each other in order to promote emergent patterns and self-organization. The graphical aspects were created to be variable, allowing us to easily change and tweak the intensity of the forces being applied as well as other visual variables such as the colors and transparencies.

The visualizations that resulted from this part of the project showed big bursts generated by the people entering and leaving taxis being affected by the events in nearby areas, creating strong and distinctive flows of people going and leaving the events (Fig. 4.2 and Fig. 4.3). While the model and representations were created based on the types of data we have available, we had to plan the interactions between the various elements and then visualize how the integration of these elements with the data responded to the behaviors we defined. We generated the graphical waves we had intended, despite not have explicitly programmed each one of them, and we were able to visualize organic flows when observing the taxi pick-ups in relation to the events despite that not being one of our intended behaviors.

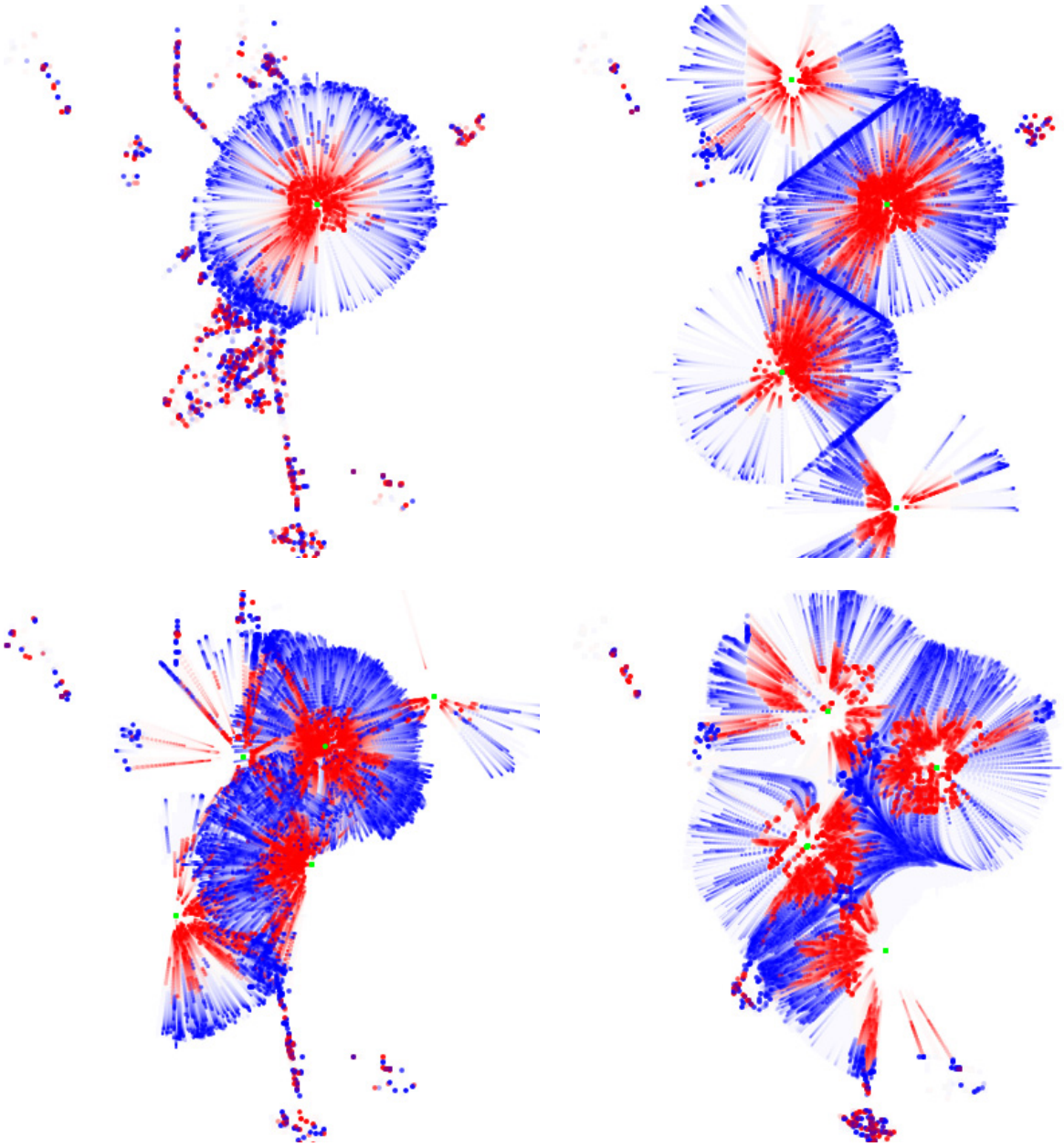


**Fig. 4.1** Various images showing early representations of the wave concept using test event points. Taxi pick-ups are represented in blue, drop-offs are in red, and events are in green. These images show the development of various applications of the repelling forces, as the pick-up locations move away from the events: the top-right image shows the event repelling forces acting only the over the closest pick-ups; the lower-left image applies the same method, but uses the initial position of the pick-ups; the lower-right image shows the function which was used in the final visualizations, where all events are used to calculate the force and direction, which generates a more organic flow and movement.

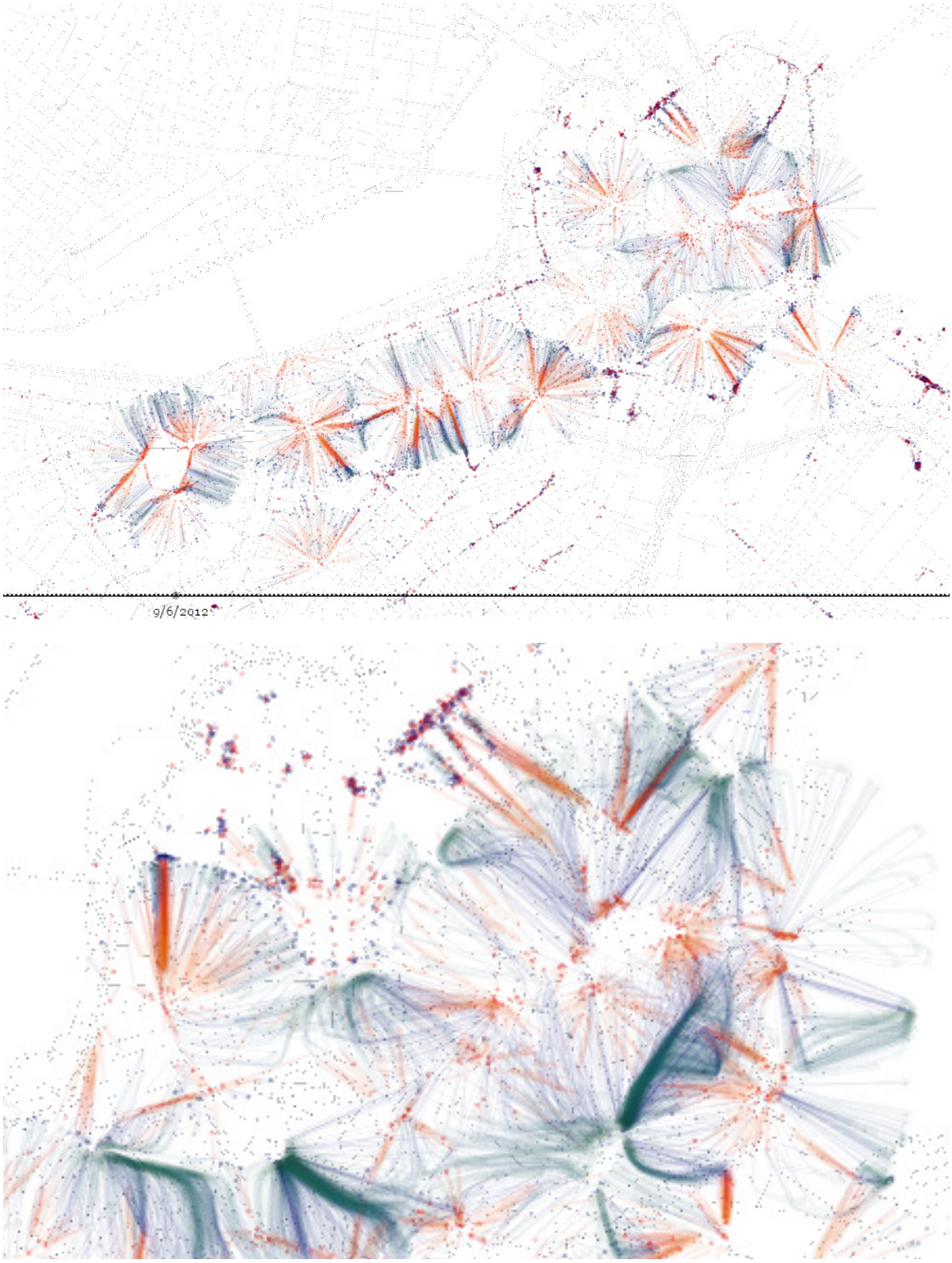



**Fig. 4.2** A representation of the data on a map, now using real event locations and times. The points now have colorful gradient trails which are more visually appealing, distinctive, and make their direction more apparent. The interactive timeline is shown at the bottom, still at an early stage.

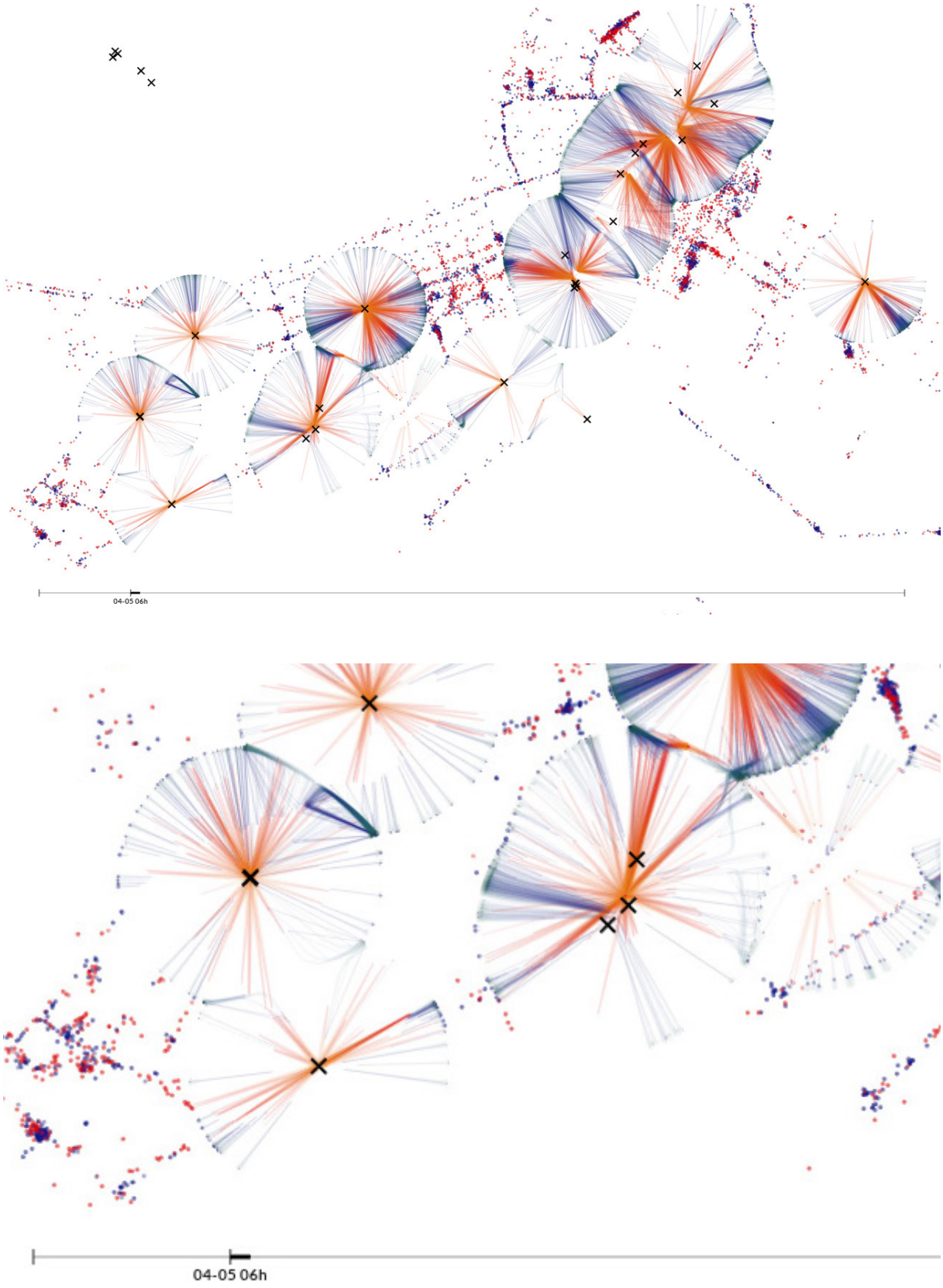



**Fig. 4.3** A visualization frame of the project much more advanced in it's development, using with graphic representations closer to their final forms. The final form of the interactive timeline is shown at the bottom which now also shows the temporal range of the data being represented, indicated by the black bar that follows the current position indicator notch.

User interaction consists of a scroll bar at the bottom which represents the timeline, where the user can select any point within the set interval of time (the month of May 2012). This consisted of a line with a small movable segment, from left to right, and a text box below it which indicates the current time selected. Moving the mouse cursor over any point in the timeline indicates the time corresponding to the point of the timeline the cursor was on with another textbox, this time above, and clicking on this point would bring the previous segment to that location, and update the visualization accordingly. There is also a time window that can be defined, which represents the range of the current point in time, meaning the visualization shows the events and taxi activities that exist within that time window, and once they move past it, they fade away. The map itself also was interactive, allowing the user to move it around and zoom in and out. Another concept we also worked with were time waves, which consisted of circles centered on events with a radius that decreases and increases as the event nears to a start or to an end. Any taxi drop off within the circle radius as the event starts is considered to be a possible attendee, while taxi pick ups within the time wave as the event draws to a close will be considered as a possible person leaving that event.

The visualizations that we could generate would depend greatly on the values we defined for certain factors like the temporal range of the taxi data being displayed, and the strength of the forces which is applied to the elements which represented that data, but we will focus on the two main final visualizations[1]. The first (Fig. 4.4) included a new element which consisted of an arc representing a direct prediction of the area where people went after being picked up by a taxi at an event – green arc – or the area where they came from to get to the event – red arc. The force based layout is still in effect, although the arcs can point out more precise locations from further away. The second visualization (Fig. 4.5) does not include the arcs (for simplicity reasons) and presents only the pick-ups and drop-offs interacting with the event locations as described previously, although the strength of the forces affecting the elements has been increased significantly. This resulted in a much more exaggerated but expressive visual effect.

Regarding the contributions of each member, some initial data parsing and representations were created by me which allowed us to make a few conclusions regarding the direction of the project which then lead to the base concept of the project, which was devised by Evgheni Poliscius. I then started implementing the engine of the program in Processing, initially using experimental values in order to test the force based layout (Fig. 4.1). Evgheni restructured the code in order to utilize the data retrieved from the files, along with the addition of an interactive map of Boston. I proceeded added my engine on to the code and created a scrollable timeline which provided visual feedback for the current time being represented, and allowed users to control the point in time they wanted to view (Fig. 4.2). After this the intended representations and functions were finalized, including the timeline and force based layout, which required the efforts of both me and Evgheni (Fig. 4.3). The final additional elements – the time

[1] Final visualizations video: https://www.dropbox.com/s/b3kd7805qq6sngo/waveModelsBeta.mp4

waves and arcs found in the first final visualization – were handled by Evgheni (Fig. 4.4 and Fig. 4.5).

In terms of future work, there are improvements that can be made, such as on the level of user interaction and graphical representations in order to present more information about the events and predictions, and even give the user more options over the data such as filtering specific types of events. A more elaborate function was the adaptation of the forces with the city's roads, meaning that as the taxi pick ups and drop offs moved towards or away from the events they would travel alongside the roads which would have resulted in much more realistic movements throughout the city.

The Big Data Challenge project allowed us to explore a lot of the process and behaviors that we also intend to investigate during the project proposed for this dissertation, from dynamic interactions to the creation of models with elements that present behaviors not explicitly programmed into them that reveal patterns in the data, though in this case it was applied to a spatial structure. Some of these methods and functions were also similarly used in this dissertation's project, despite being build from the ground up in order for them to work within the different context and new datasets, and the realization of this project proved invaluable a practical learning experience.

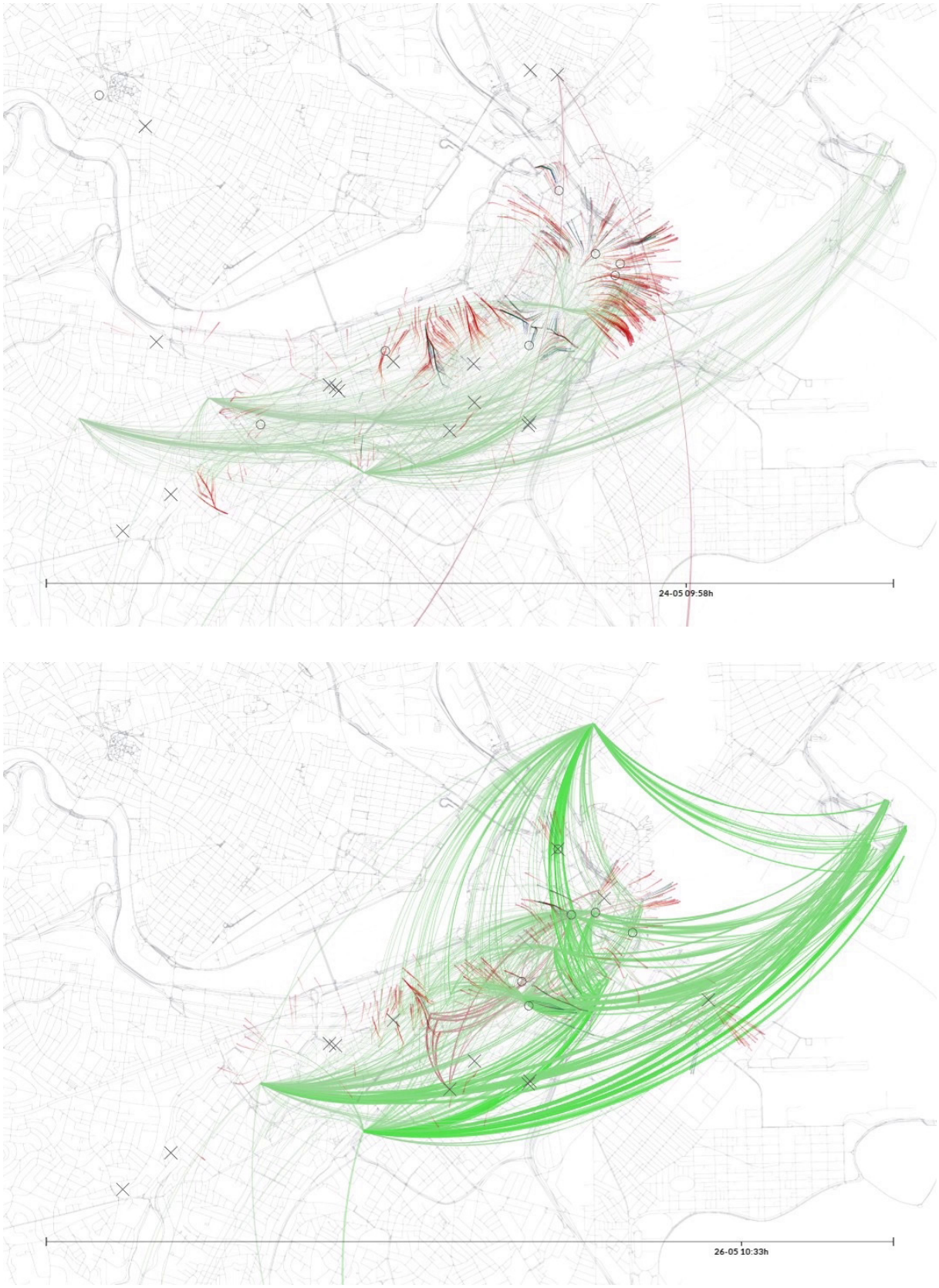
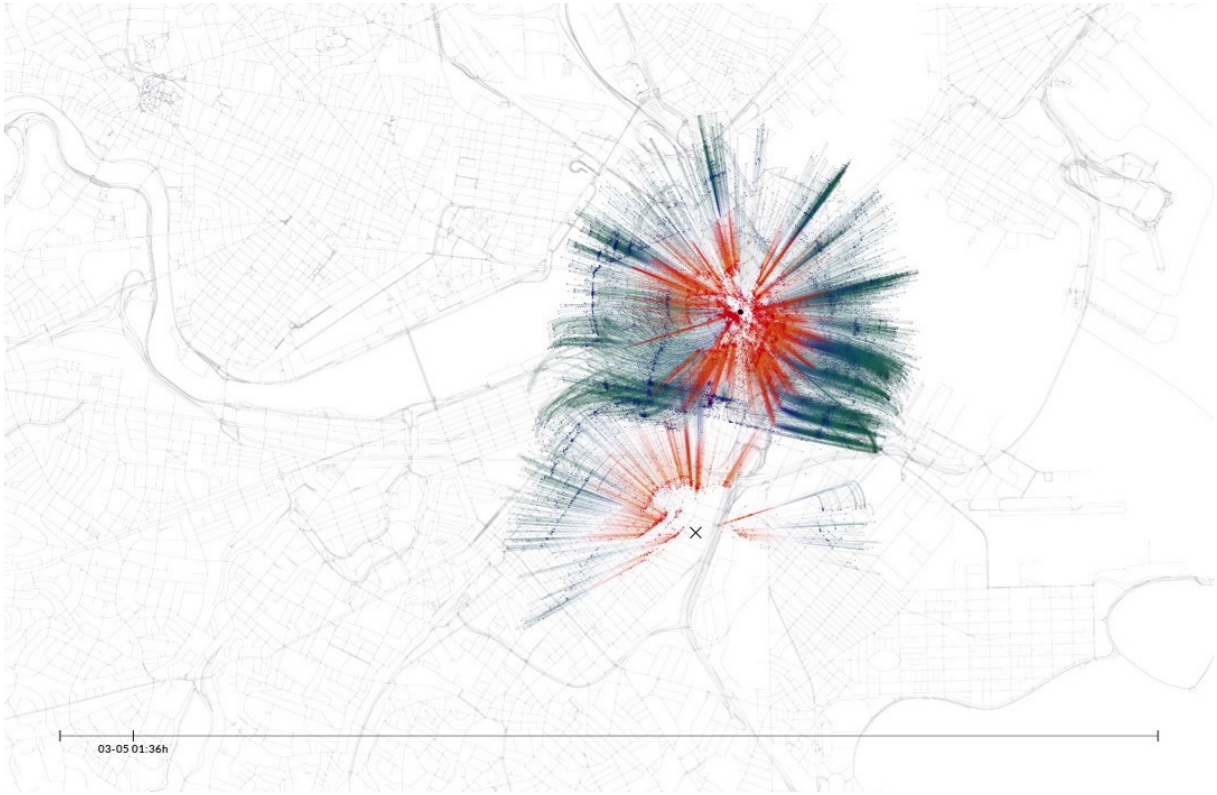



**Fig. 4.4** One of the final visualizations[2], showing both the arcs and force based layout in effect.

[2] Final visualizations video: https://www.dropbox.com/s/b3kd7805qq6sngo/waveModelsBeta.mp4


03-05 01:36h


03-05 09:52h

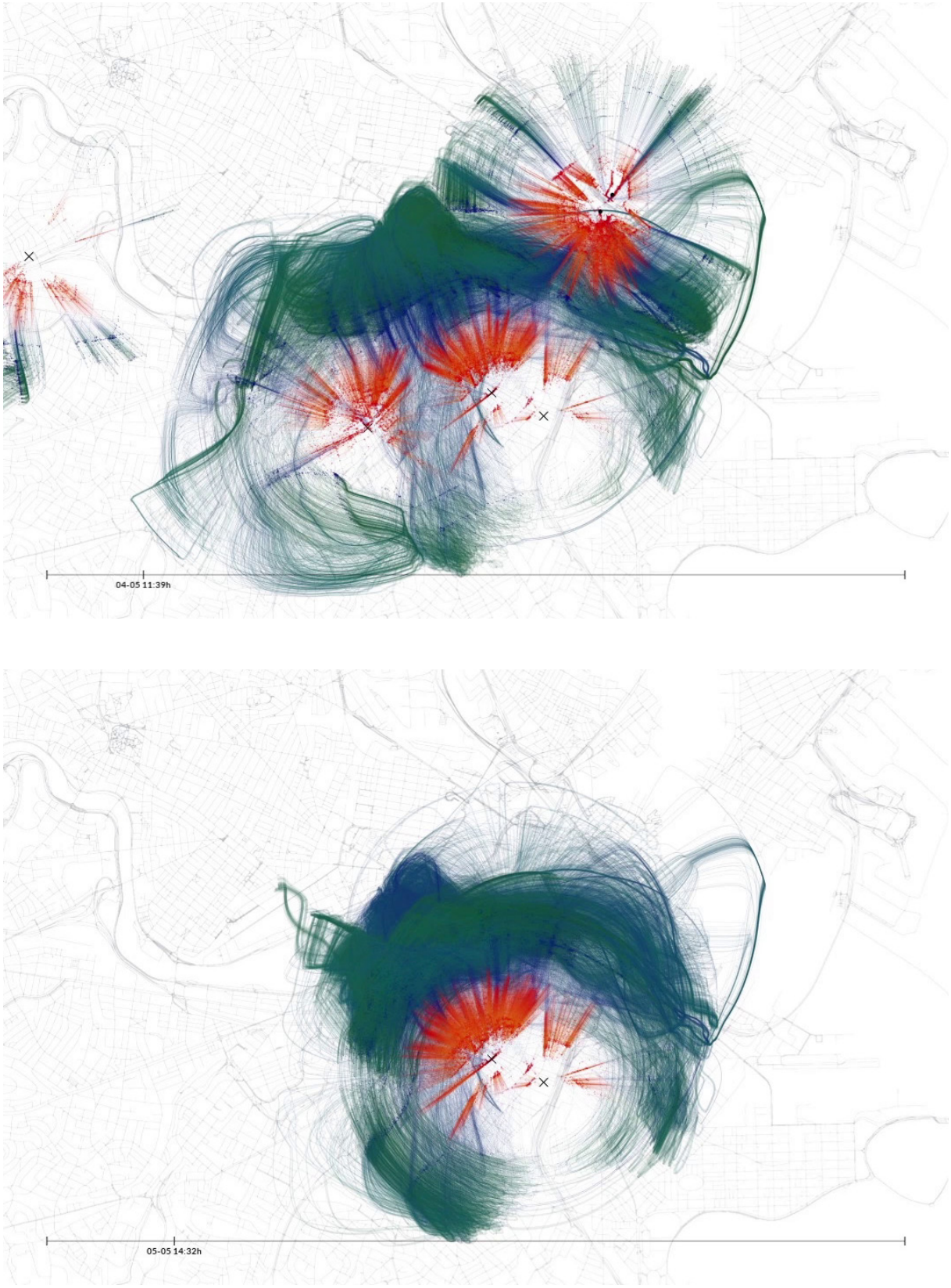



**Fig. 4.5** Various frames from one of the final visualizations[3], showing an exaggerated representation of the taxi activities reacting to the events.

[3] Final visualizations video: https://www.dropbox.com/s/b3kd7805qq6sngo/waveModelsBeta.mp4

# Chapter 5
# Visualizing genetic algorithm data

## Initial development

A genetic algorithm can be run for hundreds or even thousands generations which are, in turn, composed of hundreds of individuals, so it can be difficult to visually represent the resulting large quantities of data coherently. A dynamic approach was built in order to visualize all this information, more specifically, an application which can receive the data and output an interactive visualization which allows for multiple levels of detail as well as various layers of information which could be easily navigated and comprehended.

The application was developed with two concepts in mind: simple and minimalistic visual representations and a large focus on navigation. Due to the size and density of the data we handled it was important for it to be represented through simple elements. It is important that these are still easily identifiable, and complex elements would either suffer from loss of detail or cause visual confusion when shown simultaneously on screen. The focus on navigation is also addressing the complexity and size of the data, as interactivity allows us to manipulate the visualization's level of detail in different ways, from filters and ordering to levels of magnification. John Holland also discusses this process of selection of what to show and what to hide as a way to allow for more complex information to be communicated and understood [27].

In order to provide a general overview of the entire network of individuals and populations, the idea began with representing each individual as simply as possible, even if it meant reducing these to pixels, in which case they would have two main visual variables through which they could communicate information: position and color. Regarding position, each population of individuals would be displayed as a column, ordered by fitness, and the columns would be lined from left to right chronologically, as each population represents a generation of individuals. This way, position alone would allow the user to get a relative idea of how far an individual was in the timeline and how fit it was in relation to the rest of the population (Fig. 5.1). Color can then be used to indicate a variety of attributes, such as the actual value of fitness (instead of the relative value shown by position) or the number of descendants of that individual.  However, this was only the

original concept and, with the implementation of navigation functions, certain limitations can be bypassed such as the size restriction. Through the use of functions such as moving the graph and zooming in, it becomes possible to increase the size of the graph past the screen limits without actually losing access to the rest of the information, meaning it will not be necessary to reduce the elements to simple pixels, though this can still remain as a visualization option for the user.

Initial tests were created using randomized data that fit the parameters expected of the target dataset while the data was not available to us, which allowed for some graphical tests and some initial understanding on both the usability and the flexibility of the application. The visuals were based on the concepts described before, although the biggest focus at this point was the implementation of the navigation functions. By clicking and holding the right mouse button on screen and dragging the mouse around the user can drag the graph in any direction and visualize any part that could be off screen, but there are limits established so that the user does not drag the graph completely off screen. Using the mouse wheel the user can zoom in and out on the position where the mouse is currently located.

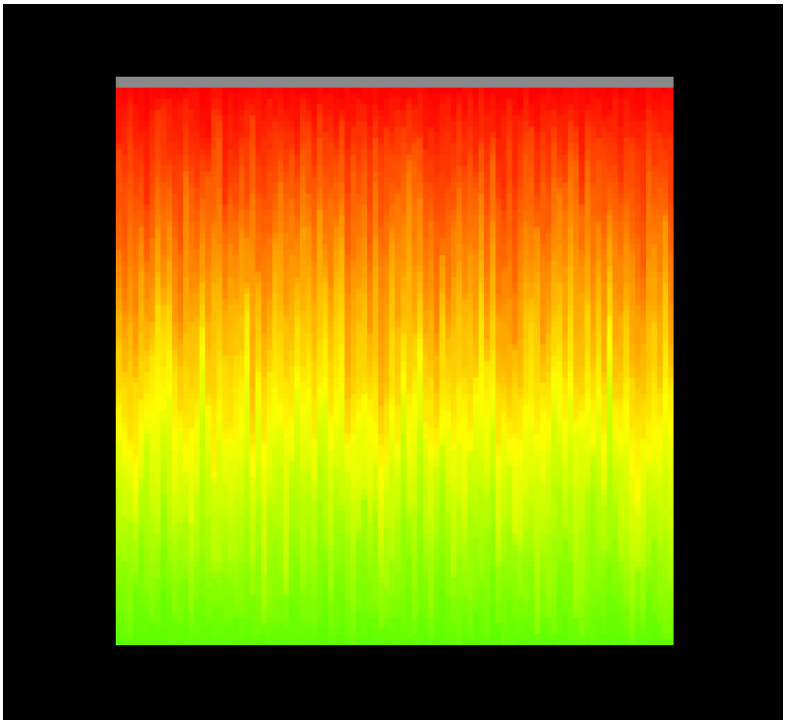


**Fig. 5.1** This screenshot shows the very first version of the application which displays the randomly generated individuals as colored squares. Each column represents ae generation, in sequence from left to right, and the individuals in each column are ordered by their fitness. Color represents the fitness of each individual, ranging from red to green. Since the data is randomized the average fitness of each column is roughly constant throughout, meaning that there are no significant increases or decreases throughout the generations.

Individuals have connections to their parents and their children, and this information allows us to build the complete family tree of any one individual, which means we can understand how the transformations that their genes underwent. By use of these connections, a function was created which allowed the user to select any individual on the graph by selecting them with the mouse and clicking them with the left mouse button, highlighting only those that are connected with it, meaning that every ancestor and descendant of the individual selected would be highlighted and easily distinguishable from the rest (Fig. 5.2). This allows the user to see both how that individual came to be, and what originated from it. If the left mouse button is pressed outside of any individual, then everything is deselected and the visualization will go back to it's default view.

An additional function allowed the selection of an entire population (a column) by selecting from an additional row of buttons placed along the top of the graph. These buttons have a different height and color from the rest of the nodes on the graph to assure that they stand out from the actual data and that there does not exist any confusion regarding this.

When a column is selected only the ancestors are shown, but not the descendants (Fig. 5.3). The reason for this was because when one selects a every individual in a population all their children would also be selected, meaning that every single individual from that population forward would be selected and no new information would actually be revealed from that selection. Furthermore, by not selecting anything forward, the selected column acts as a clear visual indicator without further need for other colored indicators. Finally, an additional advantage of this method is that it helps the visualization's processing speed by not rendering information which is already available when nothing is selected, which is every individual's connections.

With the current grid visualization the connections between each individual are not immediately perceptible, although it still serves as an overview of every individual which is connected to the selected one. In order to visualize the connections properly a new visualization mode was created in the style of a flow chart, where a variable interval is introduced between each column of the graph and the connections would be represented in that space, connecting each respective individuals with a line (Fig. 5.4). Since the interval is variable, the user can open and close the graph at will, as well as define the size of the interval, increasing or decreasing the level of detail for the representation of the connections, or removing them completely by reverting the graph to it's grid visualization.

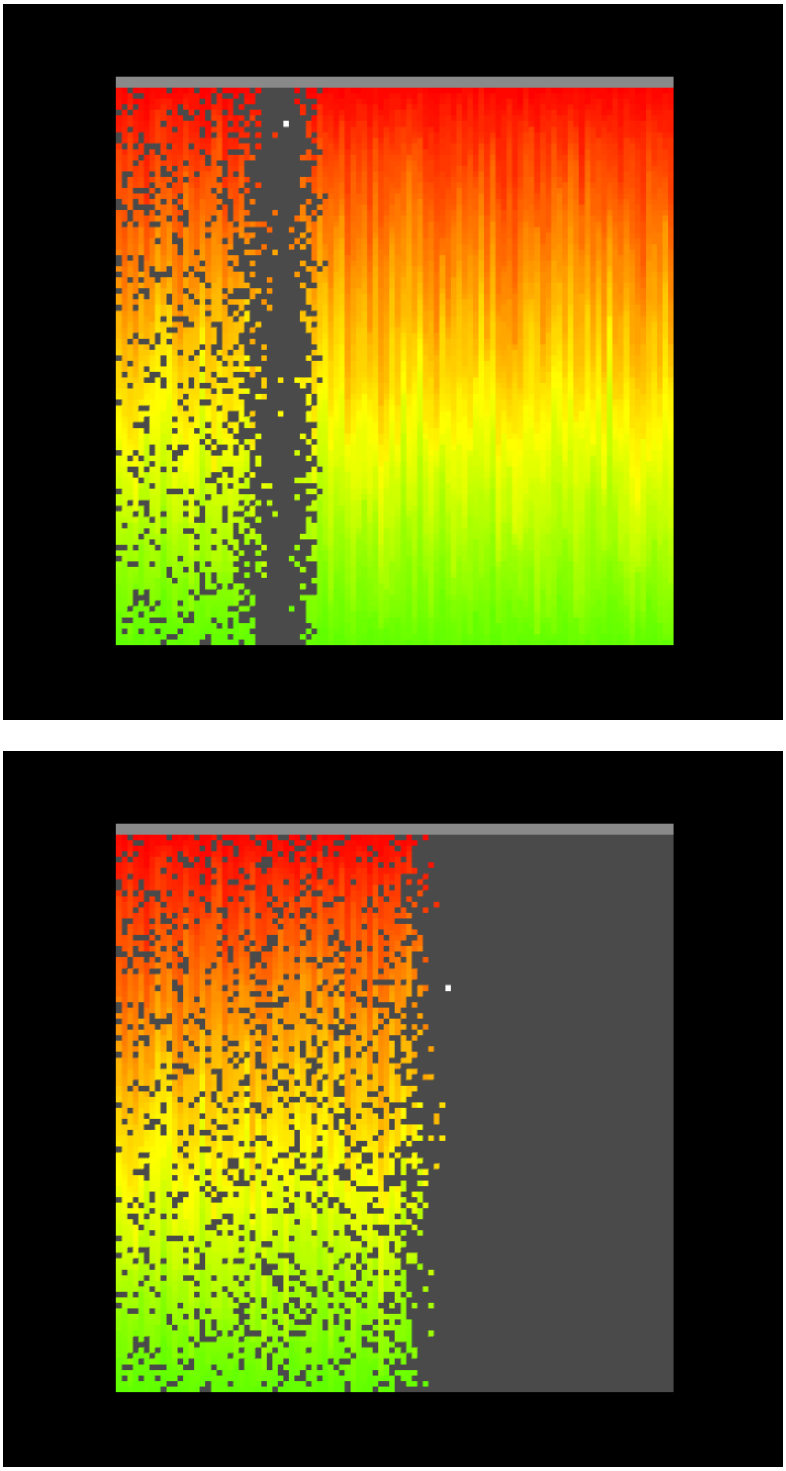


**Fig. 5.2** Screenshots depicting what happens when an individual is selected. The individual selected is represented in white, and every individual which is connected to it in some way (ancestors on the left and descendants on the right) keeps its color, while the remainder are represented in grey. In the bottom image, the selected individual has no children, meaning his family line ends with him and so every population past it is in grey.

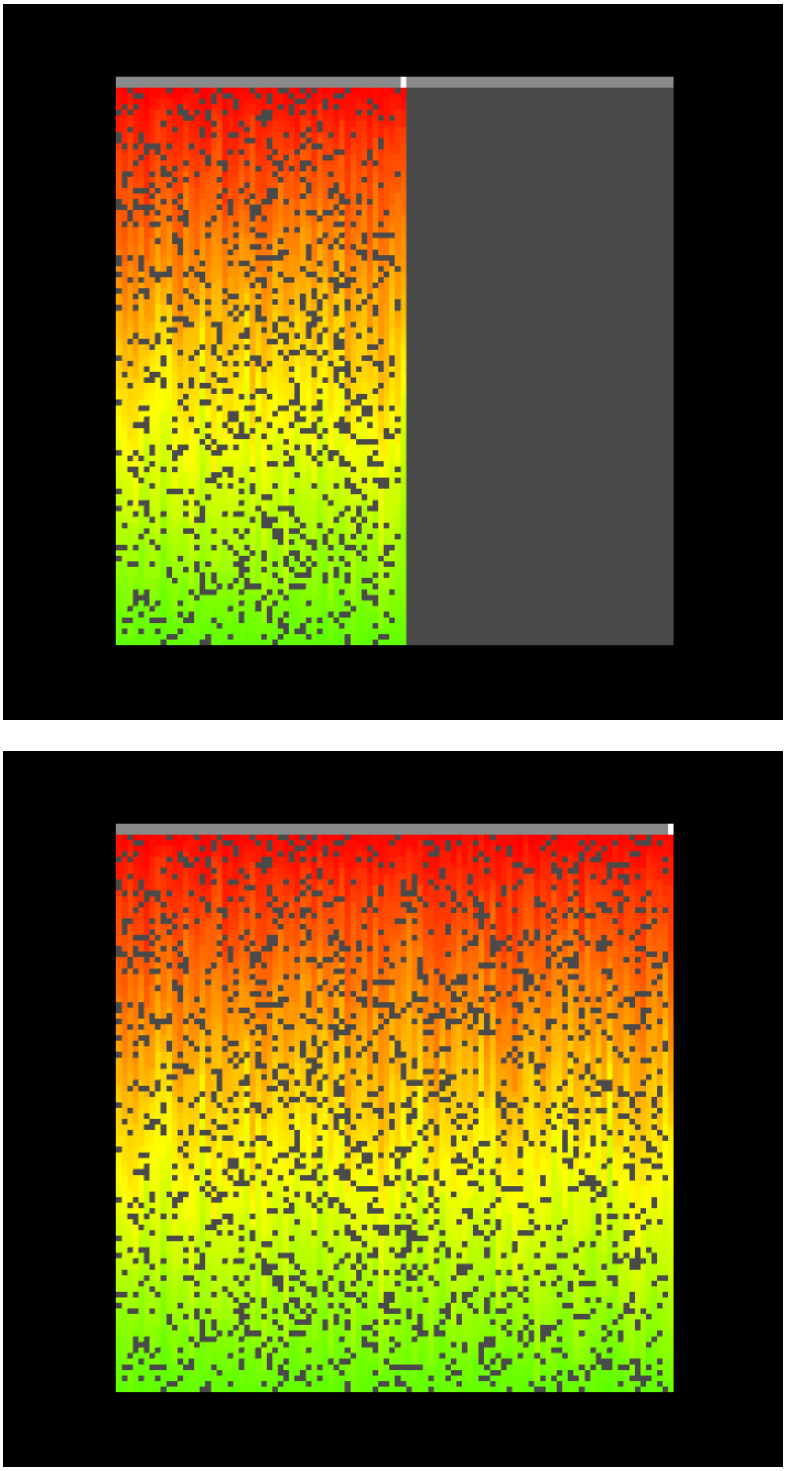


**Fig. 5.3** Screenshots showing the population selection function. The top light grey bar is divided into elements which can be selected like any other individual, and it will select the entire respective column. The bottom picture shows the last column selected, and this results in greying out every individual which is not connected to the final generation, meaning that their genes did not contribute to the creation of the final population.

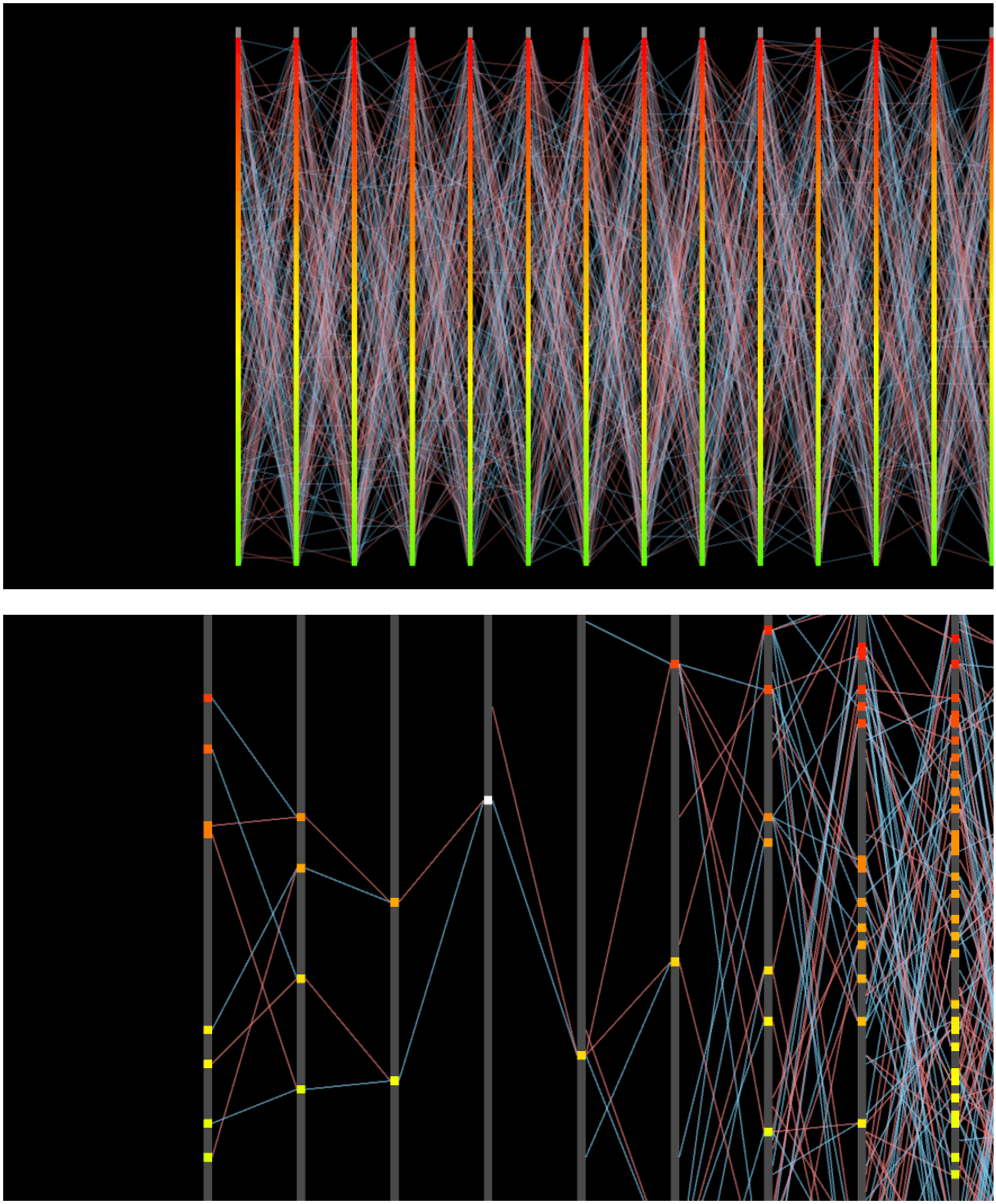


**Fig. 5.4** These screenshots show the opened graph which adds an interval between each column and connects the selected individuals which have family connections through lines. If no individual is selected then every connection is shown (top image), but with a higher transparency. The bottom image shows a close up of a selected individual, with connections to his parents to the left, and connections to his children to the right. The connections are separated into blue and red, and this is done to differentiate both parents under certain conditions, which will become more important with the actual data, as explained in the next section.

# Data description

Before we had access to the data the early prototypes allowed for some of the original concepts to be implemented which could theoretically be used to visualize general data resulting from any genetic algorithmn, as the only necessary data inputs were the individuals, their fitness and their descendents. However, it was necessary to utilize real data to properly visualize how the actual values and variations are represented and perceived, so these can be properly adjusted. The data files we received contained the data of one hundred generations of one hundred individuals from a genetic algorithmn whose purpose was to receive a curve on a two-dimensional plane and find a function which creates a curve as close as possible to the received one. This is similar to the problem described in *Chapter 2: Genetic algorithms* abeit more complex, as the algorithmn is handling individuals whose genotypes are functions instead of simple number arrays, and because these will have to represented later, it is necessary to have a basic understanding of how they work.

The genotype of each individual is actually a tree where each node represents a mathematical operation, such as an addition, multiplication or even a square root. These node make up a function that is described by following the branches in proper order. This function describes a mathematical curve which can be compared to the target function's curve in order to determine the individual's fitness. This is done by comparing the curves and not the trees themselves because the objective is to reach a function that produces a curve similar to the target function's curve, and a similar tree to the target's tree could actually wield a function with a very different curve. However, it is this tree which is subjected to mutations and crossovers in order to originate other varied solutions, and not the mathematical curve itself.

There were actually two approaches taken for the same problem: one involved the more common method that is based on natural selection, while the other based itself on *sexual selection*. The most notable distinctions between these two methods that matter to this project are that individuals on sexual selection have two chromosomes, where each contains a different genotype and is affected by a genetic operator idependently of the other, meaning that on the same individual one chromosome could be generated by a copy and the other by means of a mutation. It is also important to note that during crossovers there is now a clear distinction between the mother and the father, and this is represented in the application by coloring the connection between the mother and her child in red, and the father and his child in blue. Since there is no distinction between mother and father outside of sexual selection in the data, with the natural selection files all the connections are shown in black. The two approaches are independent of each other and are located within different data files, and as such only one type of these data files is read during each execution of the application, although multiple files of the same type of selection can still be loaded up at once.

# Interface and functions

The early prototypes allowed for some of the original concepts to be implemented but it was necessary to utilize real data to properly visualize how the actual values and variations are represented and perceived, so these can be properly adjusted. The first available data file contained the data of one hundred generations of one hundred individuals.

By visualizing this data through the current application we could already quickly discern simple patterns and tendencies, such as the evolution of the population's fitness through each generation, and how an individual's fitness influenced their chances to be chosen to pass on their genes on to the next generation (Fig. 5.5).

Along with the refinement of the visuals and navigation, new functions were implemented that would help the user find specific information. This required an interface to show all the functions available and allow the user to select which ones he wanted to execute, providing visual feedback for these choices. The interface was designed with a minimal look in mind, consisting of a black bar on the left side with small buttons that activated the various functions accompanied by a short and descriptive name (Fig. 5.7). Originally the functions which were currently not active were represented by having their respective buttons in grey, while active ones were represented with white, but was changed as the interface evolved (Fig. 5.6).

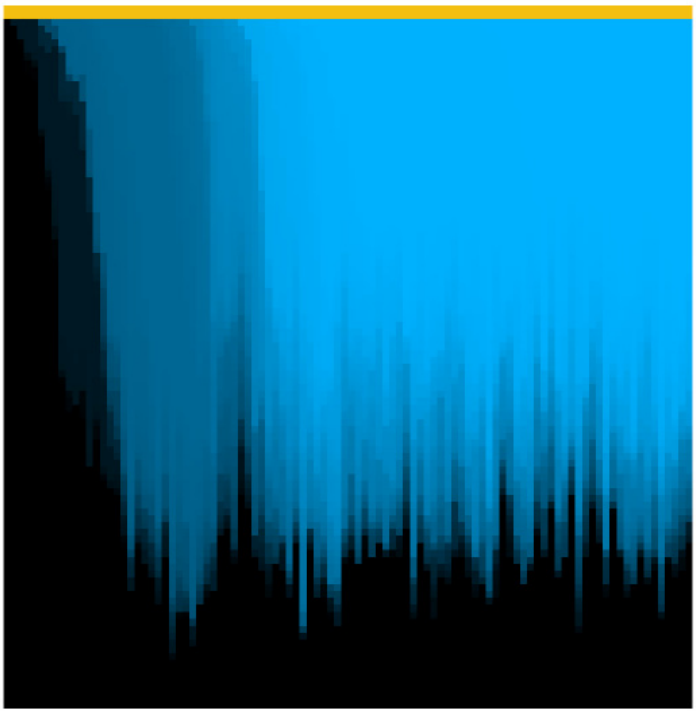


**Fig. 5.5** Final representation of a single data file, now with a white background and with the colors ranging from blue to black, where blue represents high fitness. It is possible to visualize that the initial generations had very poor individuals and that with each generation the average fitness became higher as columns on the right have brighter blues. The top bar for column selection was eventually changed to orange in order to stand out more from the rest.

Active functions were now represented in blue, while white buttons represented buttons which could be pressed but were not influencing the visualization. Grey buttons represented buttons which could not be used at that point, such as the chromosome buttons which cannot be used while visualizing natural selection data, because there is only one chromosome.

The main typography used throughout the application is condensed variant of Roboto, a sans-serif typeface family designed by Christian Robertson for Google [28]. It was a font developed to be used for the interface of smaller devices, more specifically it is the system font for the Android operating system, making it an appropriate choice for the smaller text sizes which exist in the developed interface, necessary for the optimization of space in regards to the number of functions and necessary information.

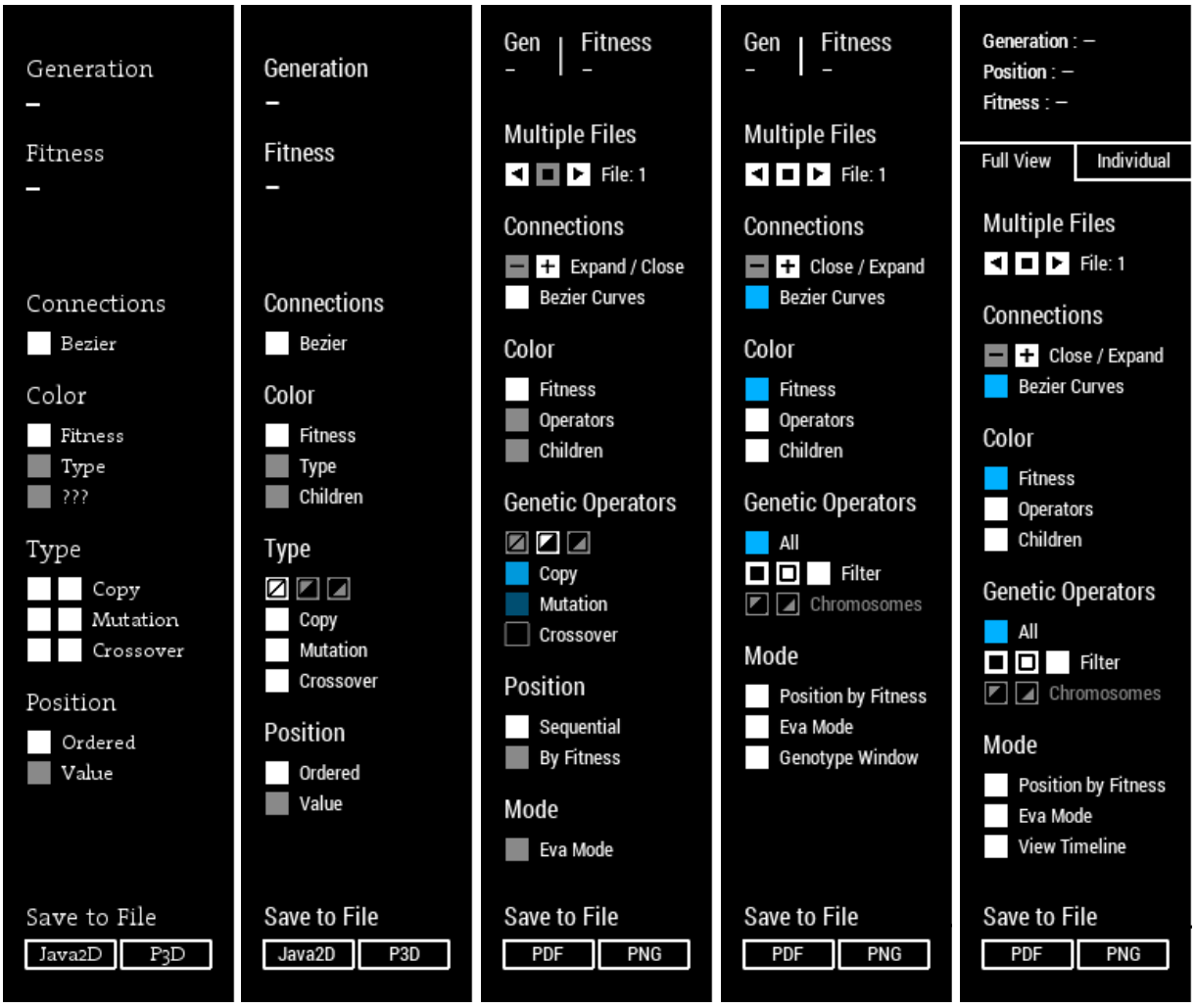



**Fig. 5.6 Select** versions of the interface during the biggest additions and changes, ordered by date, with the leftmost being the older and the rightmost being the most recent. Certain names and values would change to reflect what was being hovered on.

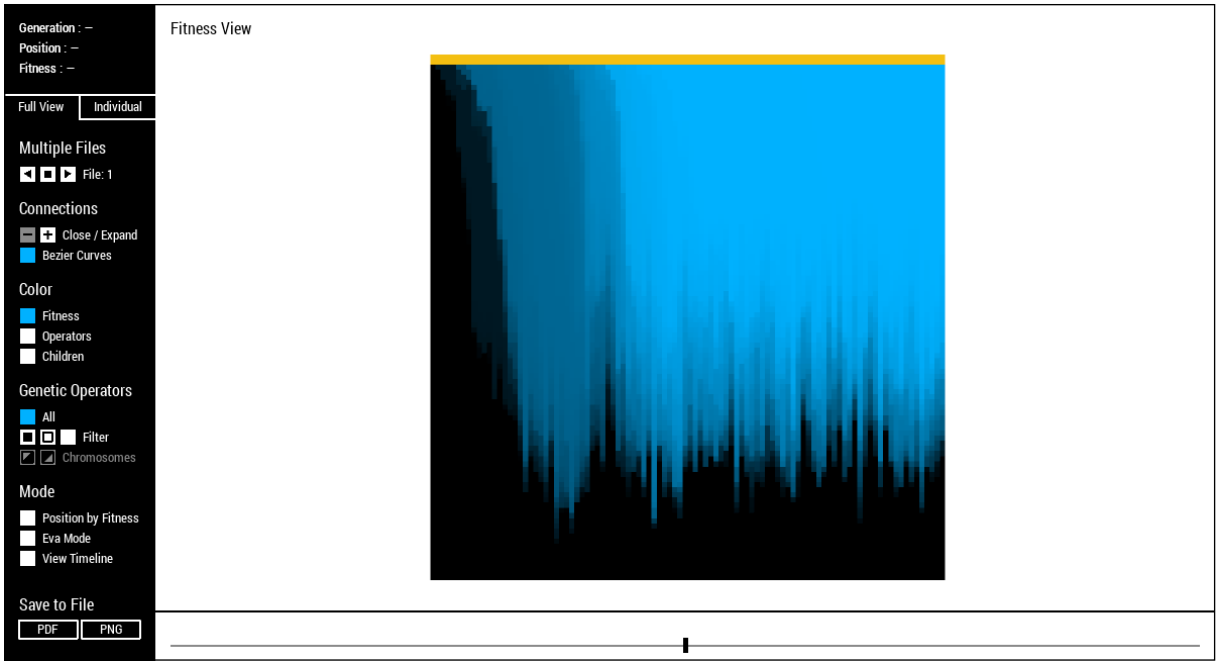



**Fig. 5.7** The most recent version of the application showing the interface on the left side, a small subtitle next to it on the top left and a scroll bar on the bottom. The main visualization in the center still uses all the navigation functions described previously.

On the top left corner of the screen there is an indicator with the currently selected individual, indicating its belonging generation, its position (meaning its fitness rank within the population) and its fitness. This value will change as the mouse hovers around the graph, showing the respective values to each individual, but when it is not hovering over anyone or when it is on the interface, it will indicate the values belonging to the currently selected individual. Below it is a tab which lets the user switch between the two main visualizations, *full view* and *individual view*, each with their own functions, representations and objectives. The current section focuses only on the full view, which is currently being represented [Fig 5.7]. On the right of this, outside of the interface bar, there is also a small textual subtitle which indicates the currently selected visualization mode, since the user has the option to change through various modes and even combine some of them.

The scroll bar on the bottom of the window is used to aid the user's navigation, mainly to help with the navigation of the open graph [Fig 5.8], allowing the user to scroll from one end of the graph to the other. This is a necessary feature because the graph can become very large horizontally and the user should not be forced to slowly drag the graph to their desired position. In addition to this, as the user drags the scroll bar there is a textual indicator with the current generation's number shown under the generation which is currently being displayed on the middle of the screen, which is also pointed out by a small arrow.

The application has various functions which were implemented to aid the user's navigation and visualization of the data[4], which are visible on the final version of the interface (Fig. 5.6, rightmost).

**Bezier curves**
This is a style option which replaces the straight lines that connect the invididuals during the open graph visualization with bezier curves [29], creating smoother visuals and a more apperent flow (Fig. 5.8). However, due to the high number of connections this setting can affect the performance speed of the application, which is why the user is given the option to toggle it.

**Close and expand**
This function is used to open and close the graph in order to visualize the connections between the individuals (Fig. 5.9), and can also be done on the keyboard, utilizing the left and right arrow keys. The user can control the width of the opening by pressing the close and expand buttons multiple times.

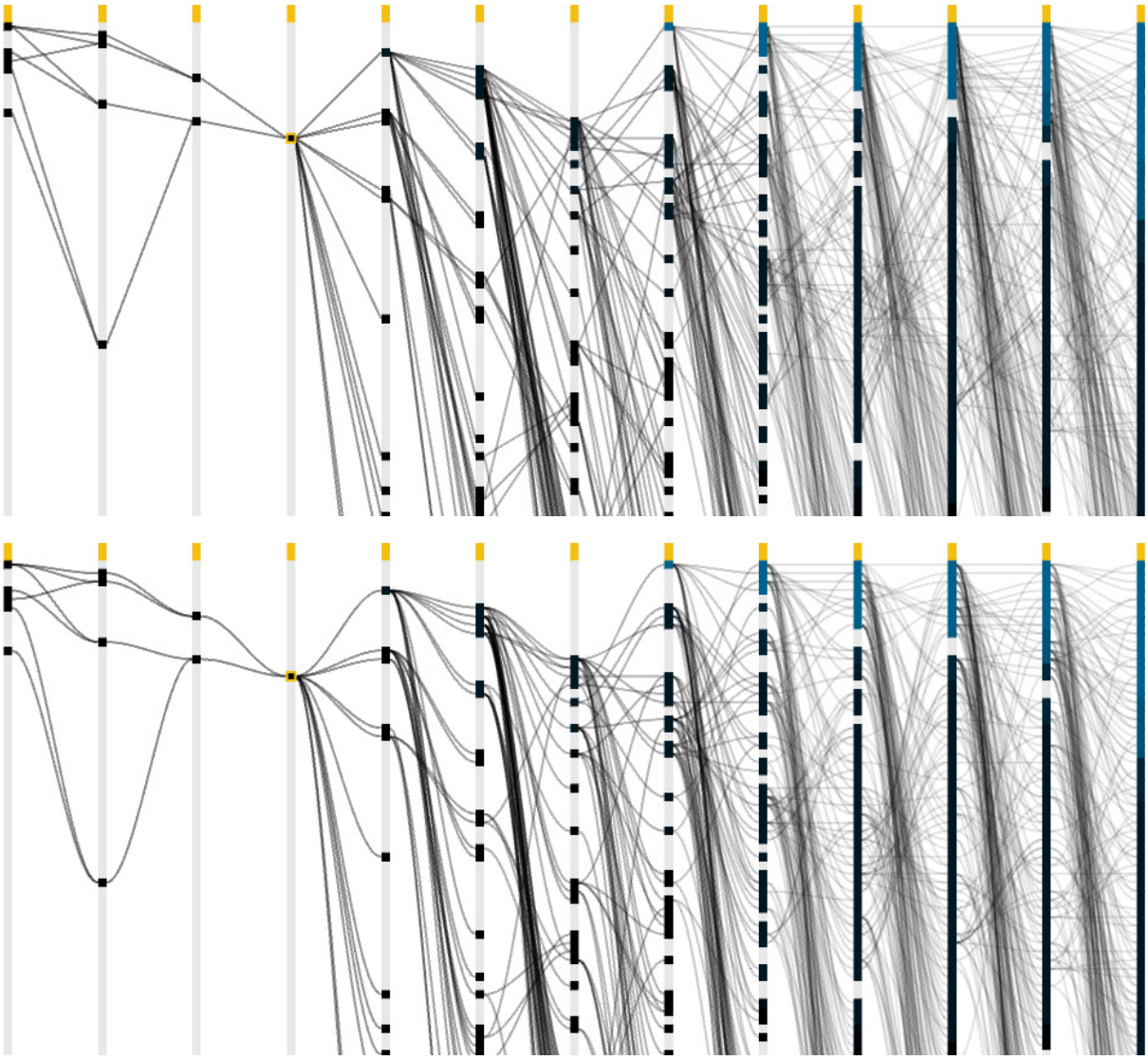


**Fig. 5.8** Two screenshots showing the difference between having bezier curves off (top) and and on (bottom).



[4] Full view video: https://vimeo.com/105060412

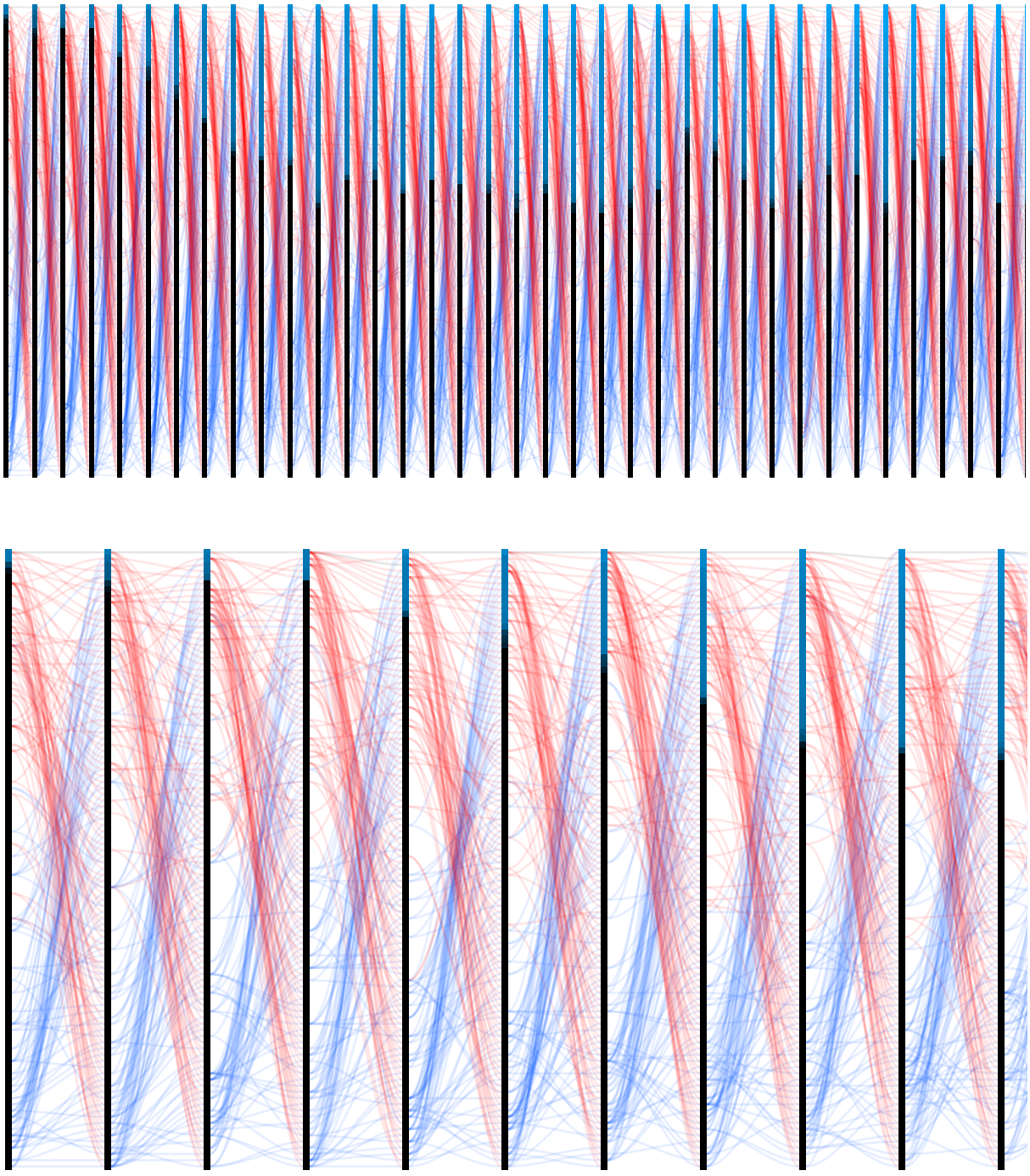


**Fig. 5.9** Screenshots showing the open graph of a sexual selection data file where the connecting lines are colored in order to distiguish the mother (red) from the father (blue) for crossovers. The interval between the columns can be controlled by the user in order to visualise the connections better, demonstrated in the bottom picture. There is a clear pattern of fit individuals having fathers with poor fitness and individuals with poor fitness having fit mothers.

**Color change**
While initially color was only used to represent the fitness value of each individual (Fig. 5.10), more options were added along with the development. The genetic operator option shows the genetic operator present at the conception of each individual, which can either be a mutation, crossover or a simple copy of another (Fig. 5.11). The children option represents the number of descendants each individual has through brightness, thus representing individuals with many children with bright colors, and using black for those with none (Fig. 5.13).

The overall color scheme also underwent a lot of changes but it was finally settled on a simple system for every color mode where blue represents high values, black represents low values, and light grey represents nodes which are not selected. The genetic operators also were distinguished with colors initially, but their representation was changed to shapes instead of using colors in order to make them more distinguishable from each other and easier to identify. An option to filter each genetic operator is also available, which removes the shapes and reduces the colors to a binary scheme, where the colored individuals will represent those with the selected genetic operator (Fig. 5.12).

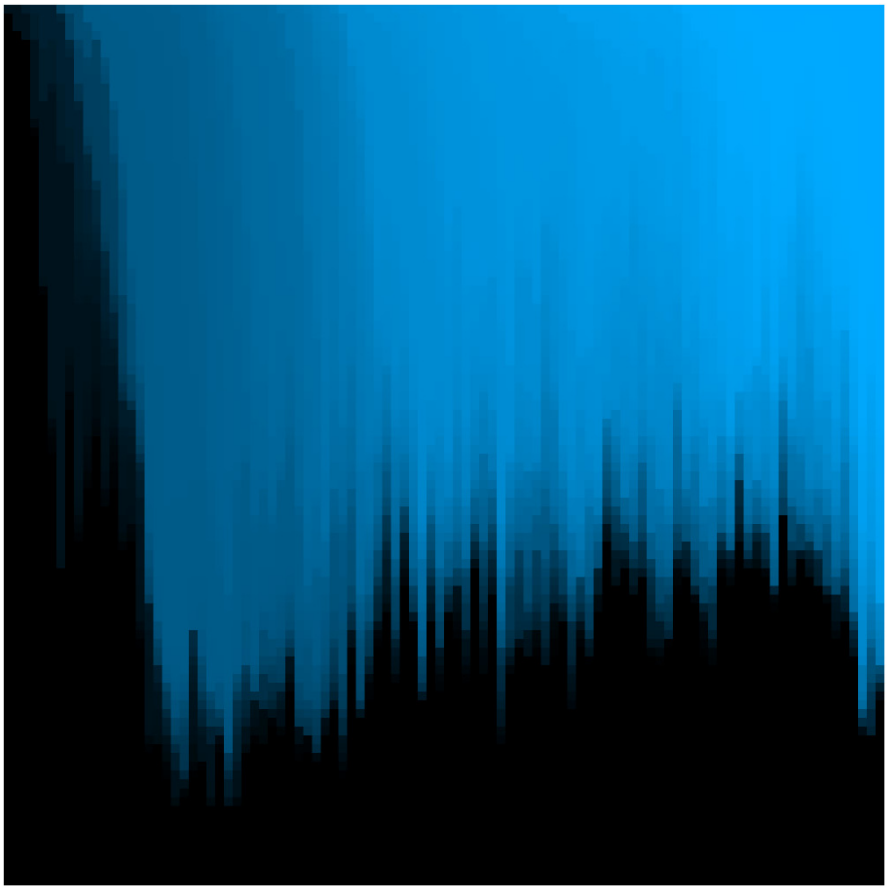


**Fig. 5.10** A data file being represented through the default color scheme which maps color to fitness.

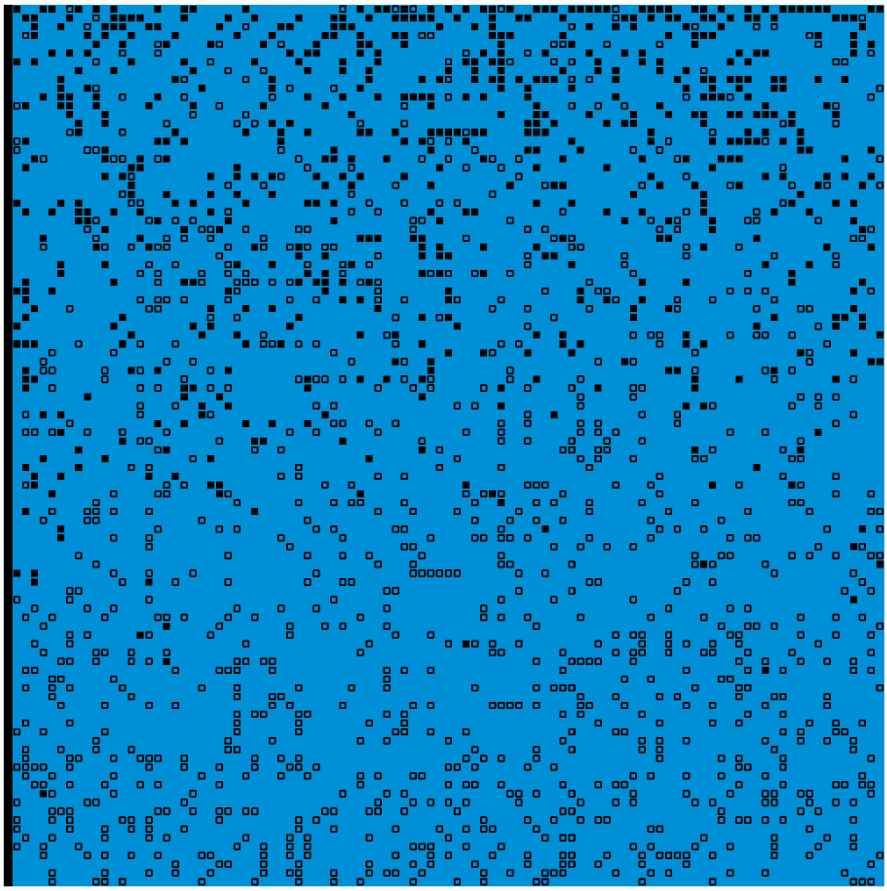


**Fig. 5.11** Same data file from Fig. 5.10 being represented through the genetic operator option. The image in the top right shows a close up of the symbols being used. Filled in squares represent copies, unfilled squares show mutations and those that are competely empty show crossovers.

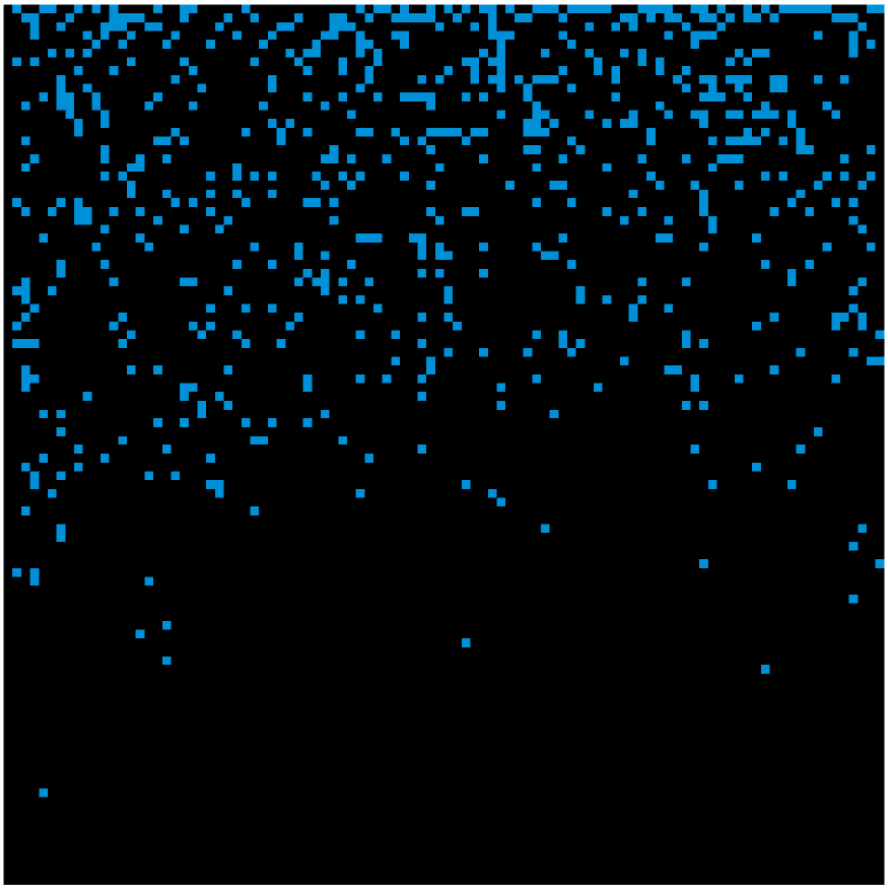


**Fig. 5.12** Same data file from Fig. 5.10 with a filter which only shows inidivduals created through copies, shown in blue. It is also possible to filter them by crossover and mutation.

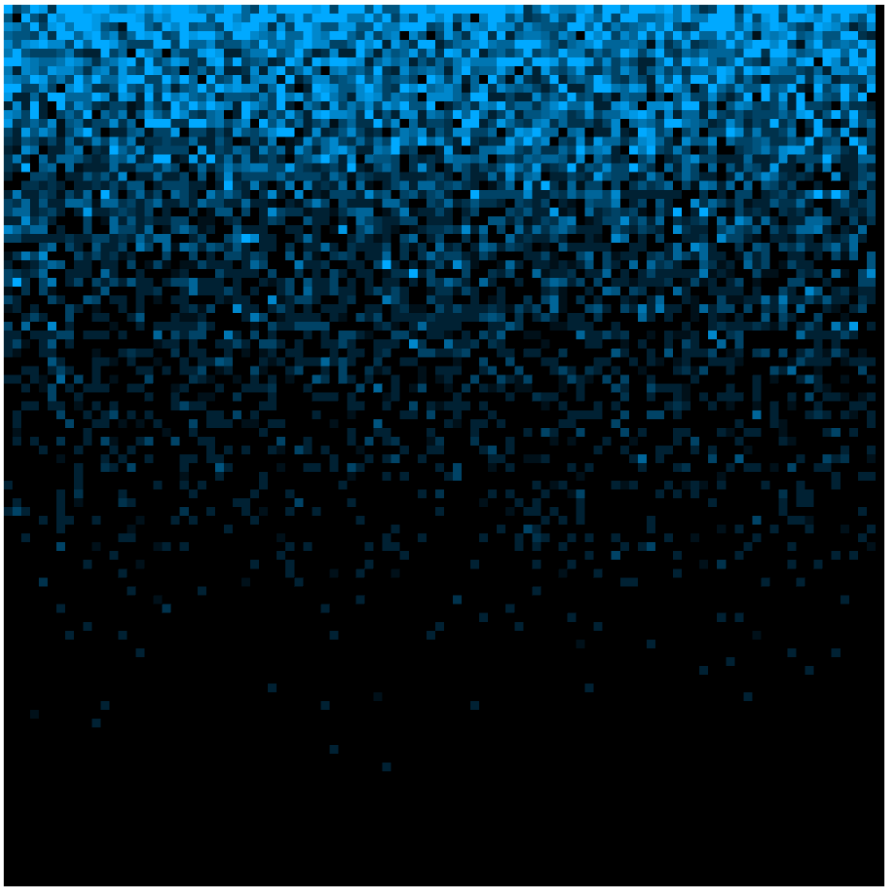


**Fig. 5.13** Same data file from Fig. 5.10 using color to represent the number of children of each individual. It is noteworthy that the most fit individuals (at the top) are the ones that originate the most children, as they are the ones chosen.

**Chromosome selection**

When simulating sexual selection the genetic operators act on each chromosome independently, meaning that an individual can been created as a result of both a crossover and a mutation. Each chromosome is represented individually and the user has the option to switch between them using the buttons on the interface. This way, one can visualize the genetic operators affecting each chromosome at a time. This option is unselectable when visualizing the natural selection data because they only contain one chromossome per individual.

**Position by fitness**

A different visualization mode was added that maps the vertical position of each individual to the value of its fitness, meaning the individuals will be placed higher the more fit they are, giving the user another way to perceive the fitness curve over the passing of the generations. (Fig. 5.14) This mode also works as an alternative to color in order to visualize the fitness growth over each generation, and allows fitness to be visualized alongside another variable mapped to the color of each individual, such as the number of children or the genetic operator.

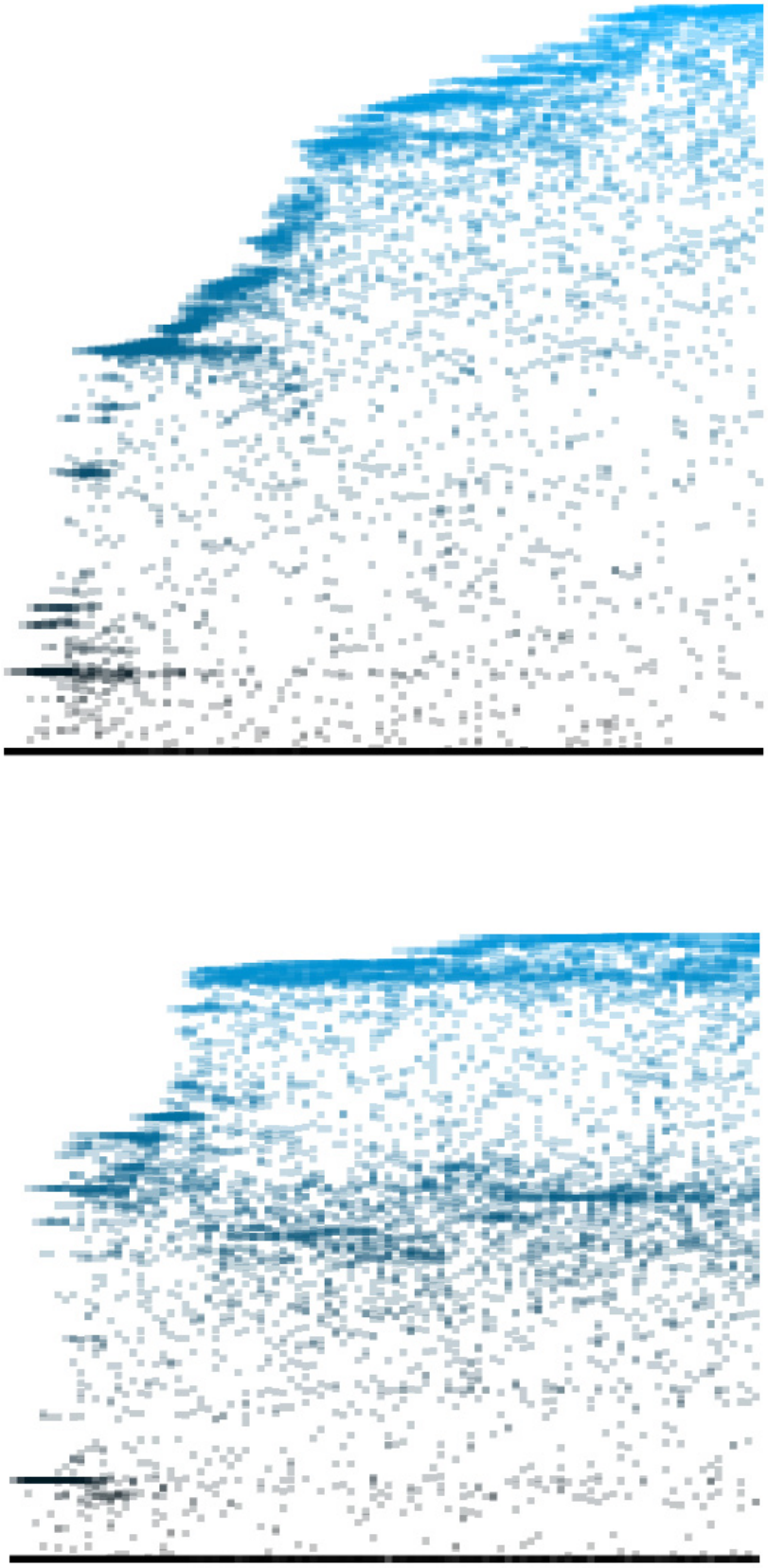


**Fig. 5.14** Two different data files shown through the mode position by fitness. It is easier to precieve the evolution of the fitness throughout the generations, as position is more discernable than color. The top image is showing the same data file from Fig. 5.10.

**Eva mode**
This mode is named after Eve, who, according to an Abrahamic religion's creation myth, is the first woman which all humans descended from. When eva mode is active the user can select an entire population in order to find the closest ancestor (or ancestors) which is related to every individual in that population (Fig. 5.15). This allows us to visualize how many generations it took for a previous individual's genes to spread scross an entire population (Fig. 5.16).

**Save to file: PNG and PDF**
In order to fully view the graph as a static image, the application allows the creation of an image file of the visualization with the current parameters, meaning the outputted image will match the colors and any other selections that are shown on screen, and its size will match the current zoom level. However, this is not a simple screenshot since the saved image will contain the entire visualization and not just what is visible on screen, without any of the interfering elements of the interface. The image can be saved as either a PNG or a PDF file.

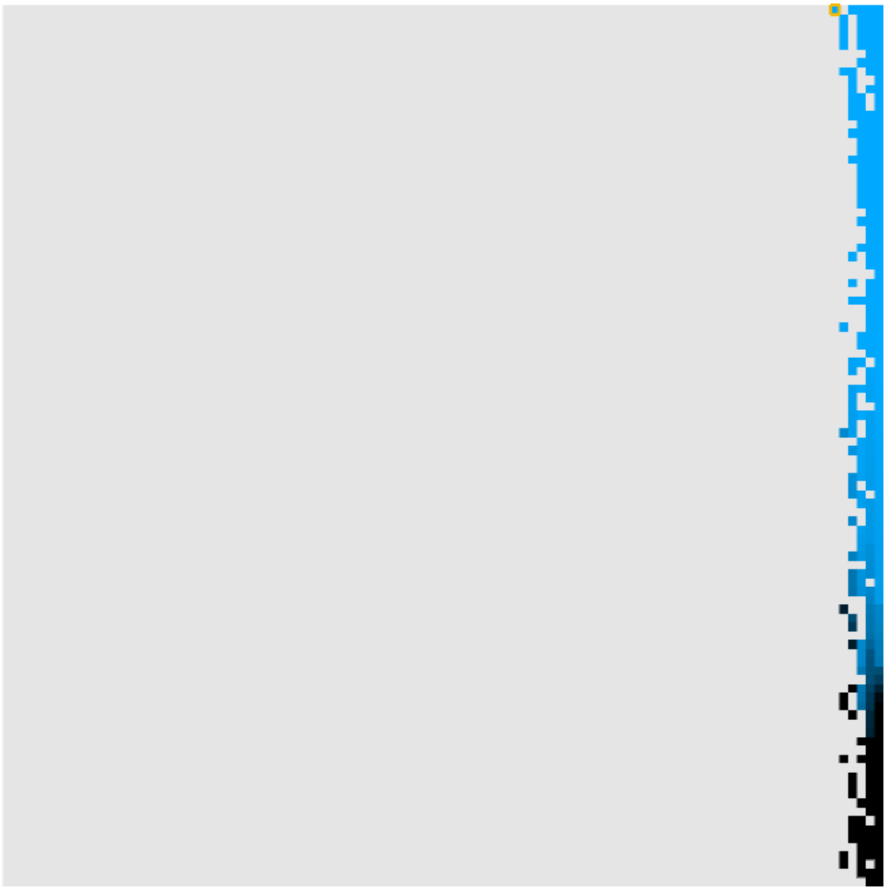


**Fig. 5.15** Screenshot showing the data file from Fig. 5.10 with the final population selected with eva mode on. The top left individual has a yellow border indicating that it was the closest individual found that is connected to every individual in the last population, which was selected. If more than one indiivdual is found in the same generation that is also connected to every individual of the selected popualtion, then they will also have a yellow border.

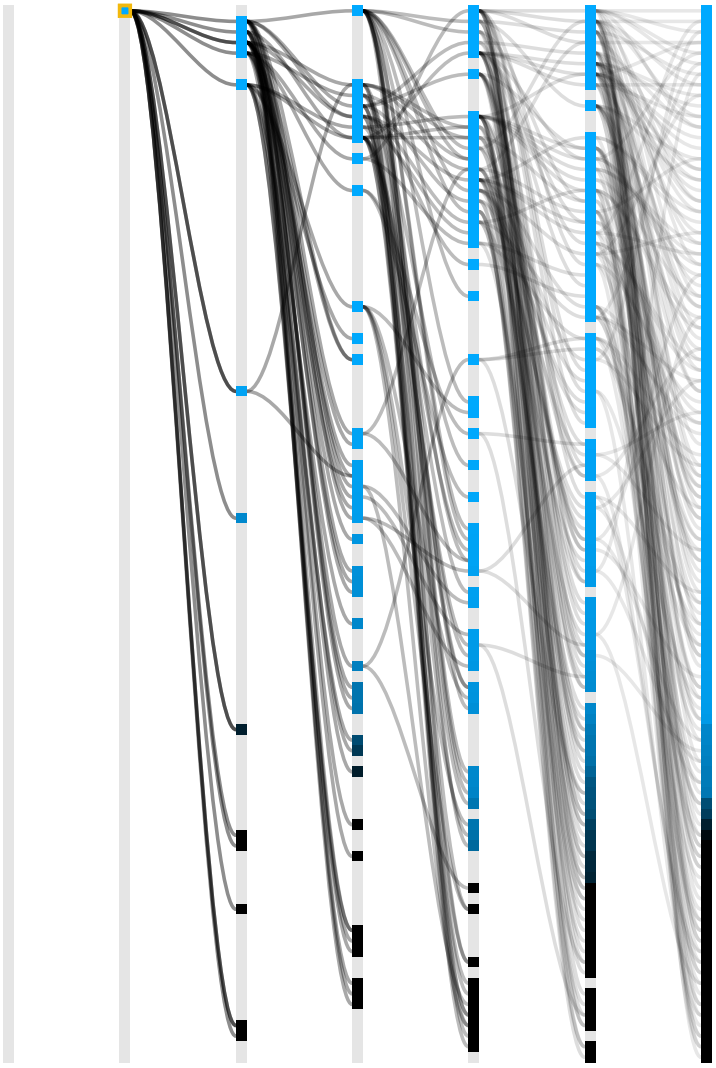


**Fig. 5.16** Screenshot showing the same file in the previous image (Fig. 5.15) but with an open graph. Through the connections it becomes more apparent how quickly a fit individual's genes can spread through an entire generation.

**Load and visualize multiple files**

Genetic algorithms rely on random values to create the initial population and maintain diversity in the subsequent generations, and, as such, each time they are run the results will be different and a single run may not yield the best results, so there will typically be multiple data files files which will need to be analysed.

The application was prepared to receive and display multiple files at once (provided they all have the same format), which let the user switch the current file being displayed through buttons located on the interface, allowing him to quickly switch between any of the files loaded on to the program (Fig. 5.17 and Fig. 5.18). When the file is switched the current settings are not reset, meaning that, for example, if a column is selected, that column will remain selected as the user changes the current file and the user can quickly visualize the same selection in the remaining files.

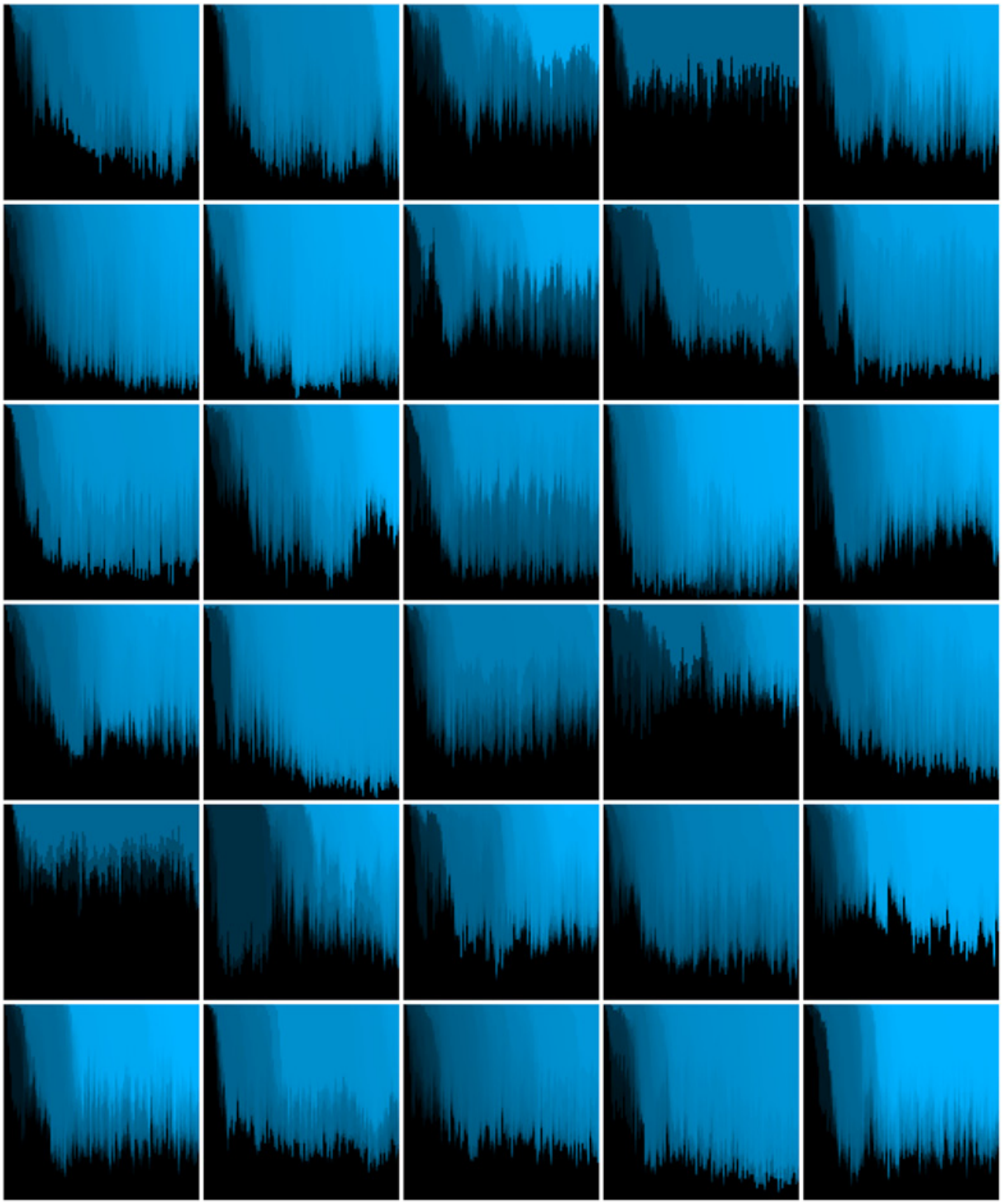


**Fig. 5.17** Screenshots of the thirty natural selection data files in the default fitness color option. The color mapping is done by using the maximum and minimum fitness values of every file, meaning the brightest blues correspond to the best individuals of all files, and not the best individuals of each individual file. As an example, the fourth file in the top line has very dark colors in comparison to many of the other files because it was not a very successful run and didn't achieve very fit individuals,

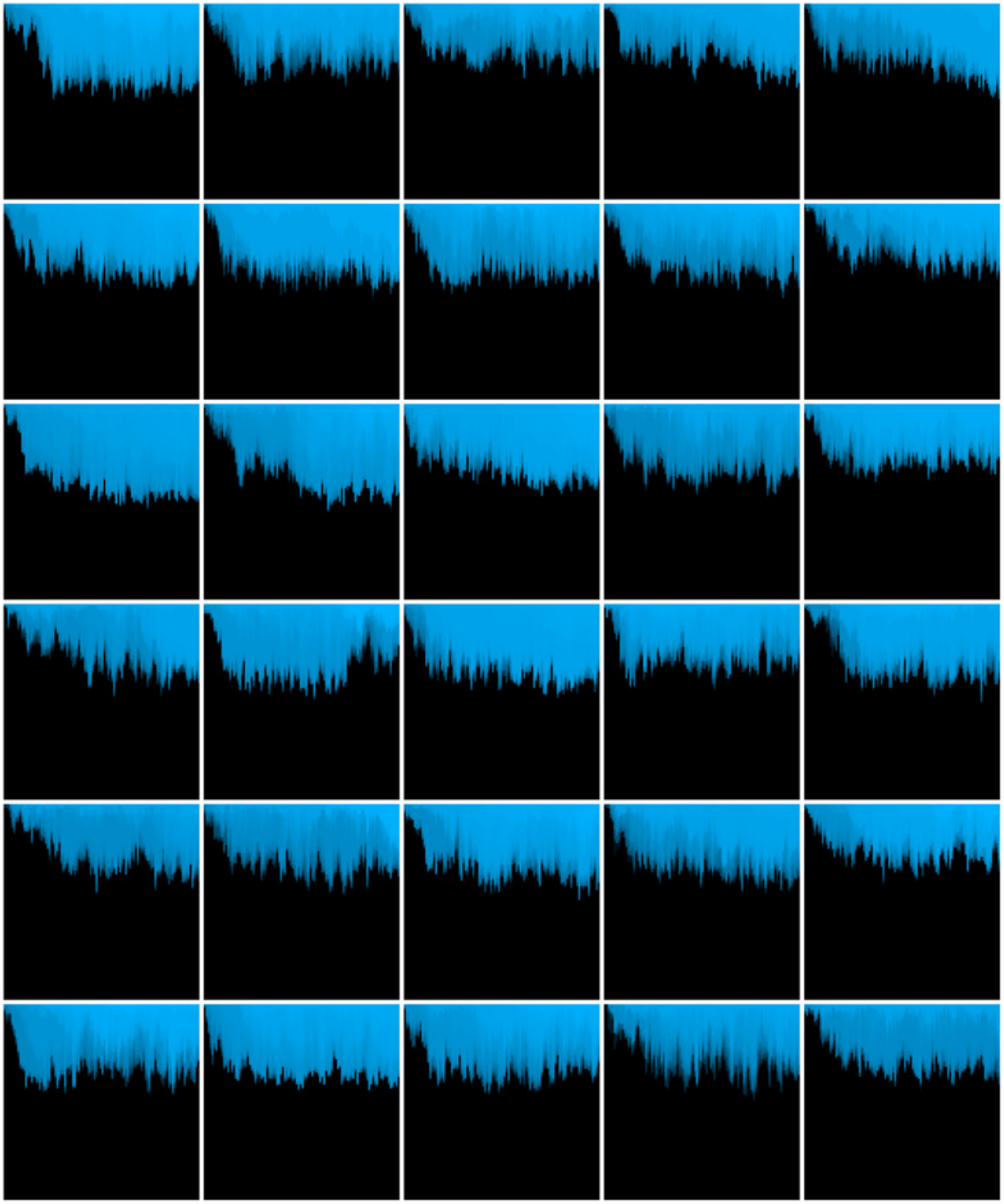


**Fig. 5.18** Screenshots of the thirty sexual selection data files in the default fitness color option. Compared to the natural selection files it is noticeable that there are a lot of less fit individuals throughout, represented by the lower half being persistently in black.

Because the number of files can be rather large to analise individually, a function was implemented to create a graph which represented the average of all the values in every file[5], activated by pressing the button between the switch file buttons. The fitness value, genetic operator selected and number of children was averaged for each position across every file and the displayed with the same method as any other file, with the exception of the genetic operators where the amount of each operator chosen for each position across the files decided the transparency of each genetic operator symbol, which were then overlayed (Fig. 5.19, Fig. 5.20 and Fig. 5.21). The number of operators chosen is hard to identify based on this overlay, however it still is possible to identify the most common genetic operator in each position, and the user can still use the filter options as before. The generated images can mainly be used for identifying trends across a large number of files.

Selecting an individual or population works like before, except every individual connected to that position across every file will be selected, and the less common the connections, the more transparent they will appear in order to show the distribution of connections across every file (Fig. 5.22). The opened graph visualization will also display every connection across every file. Other modes will also adapt to this new graph: eva mode will show the closest ancestor of every file from a selected population (Fig. 5.23), and the position by fitness view will display the individuals based on their averaged fitnesses (Fig. 5.24).

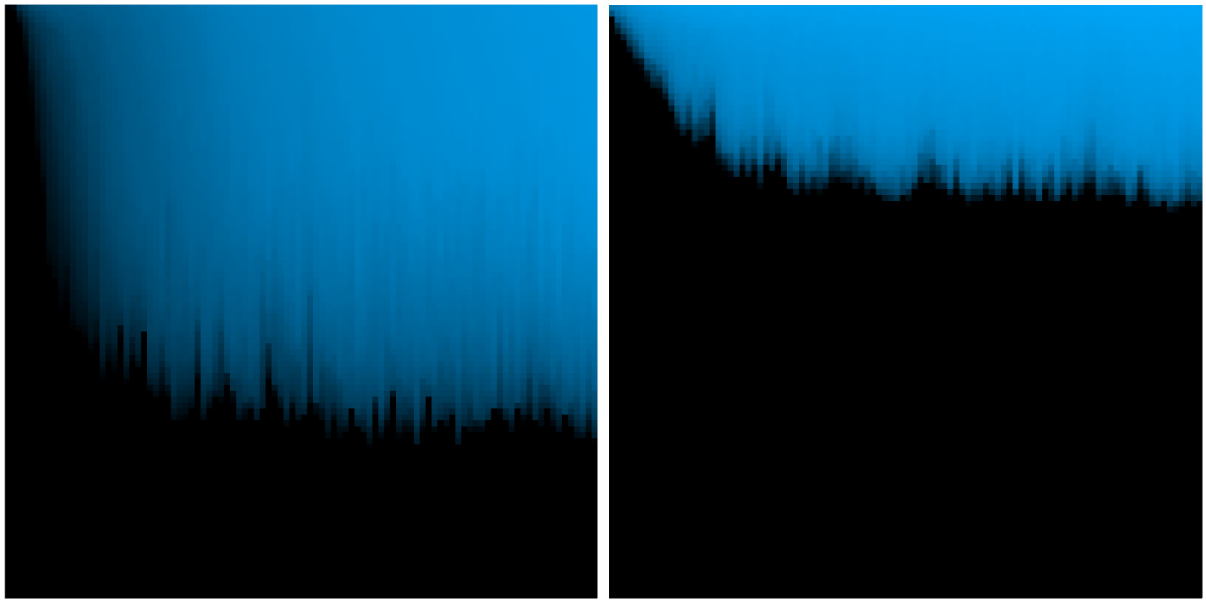


**Fig. 5.19** Screenshots of the thirty file average of the fitness of both the natural selection (left) and sexual selection (right) data.



[5] File average video: https://vimeo.com/105063501

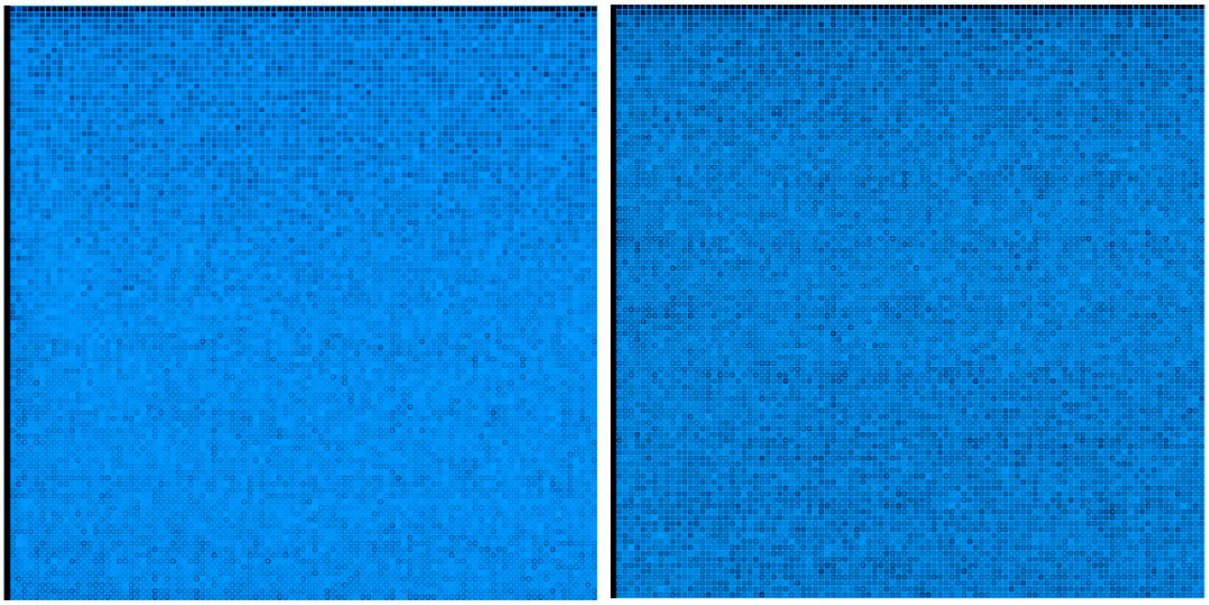


**Fig. 5.20** Screenshots of the thirty file average of the genetic operators of both the natural selection (left) and sexual selection (right) data. Even through detailed analysis it is hard to discern large patterns, though in the natural selection data the top half is darker, indicating that the most fit inidividuals are more often copied, while in sexual selection the choice of genetic operators appears to be more uniform throughout.

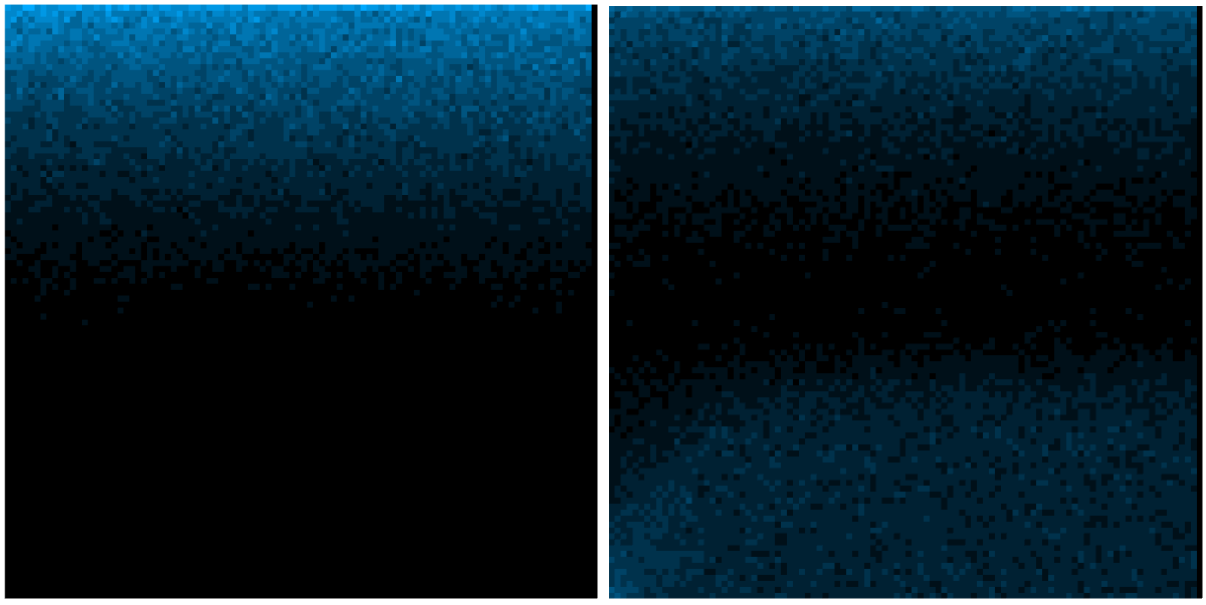


**Fig. 5.21** Screenshots of the thirty file average of the number of children of both the natural selection (left) and sexual selection (right) data. It is quite discernable that in sexual selection both the best and the worst individuals are more often chosen to create children, unlike in the natural selection data, where only the most apt are picked.

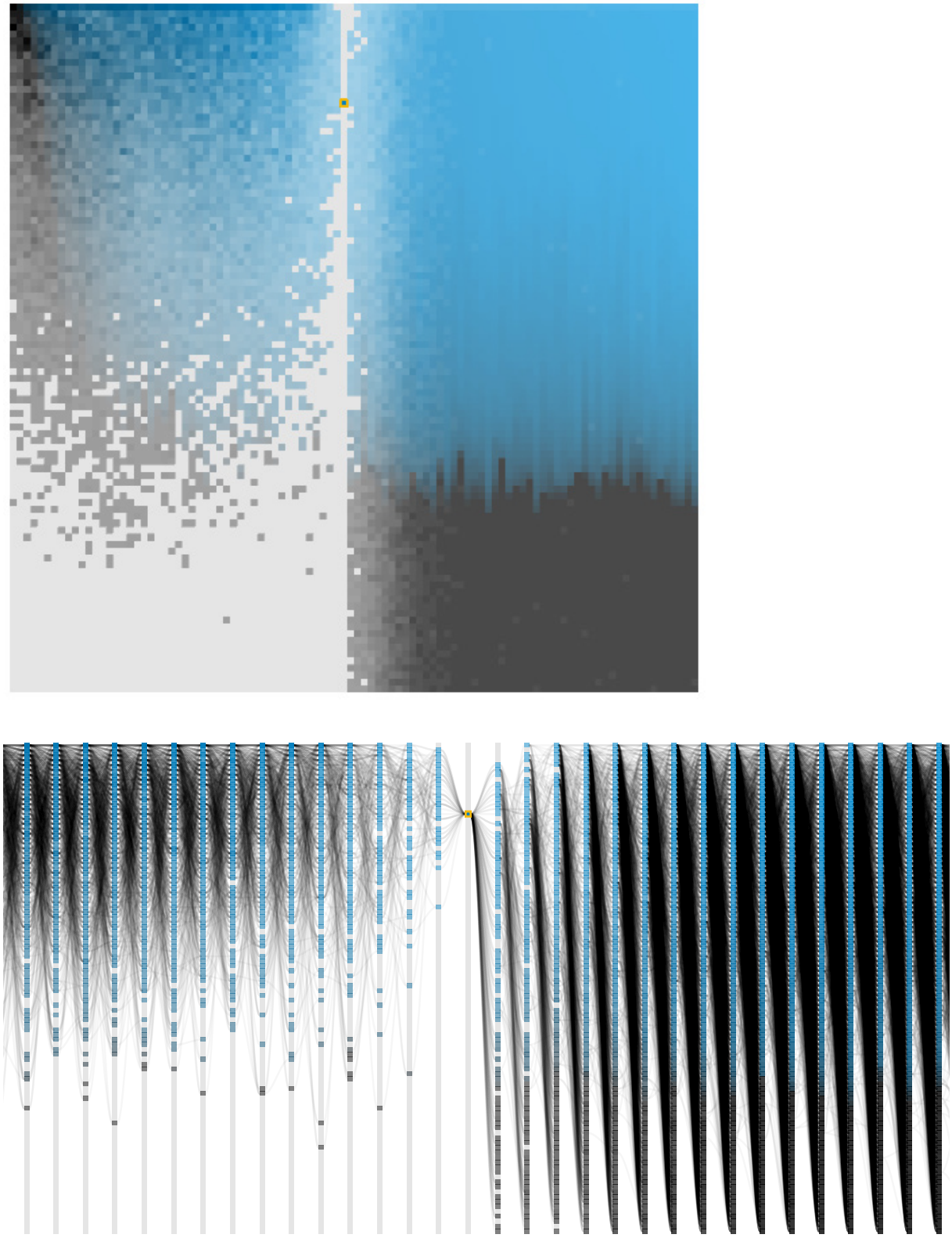


**Fig. 5.22** Screenshot of an individual being selected in the file average view. This will "activate" every individual that is connected to that psoition in every file and the more common individuals that are connected, the less transparent each position will be. The bottom image shows part of the top graph opened which reveals every connection across every file.

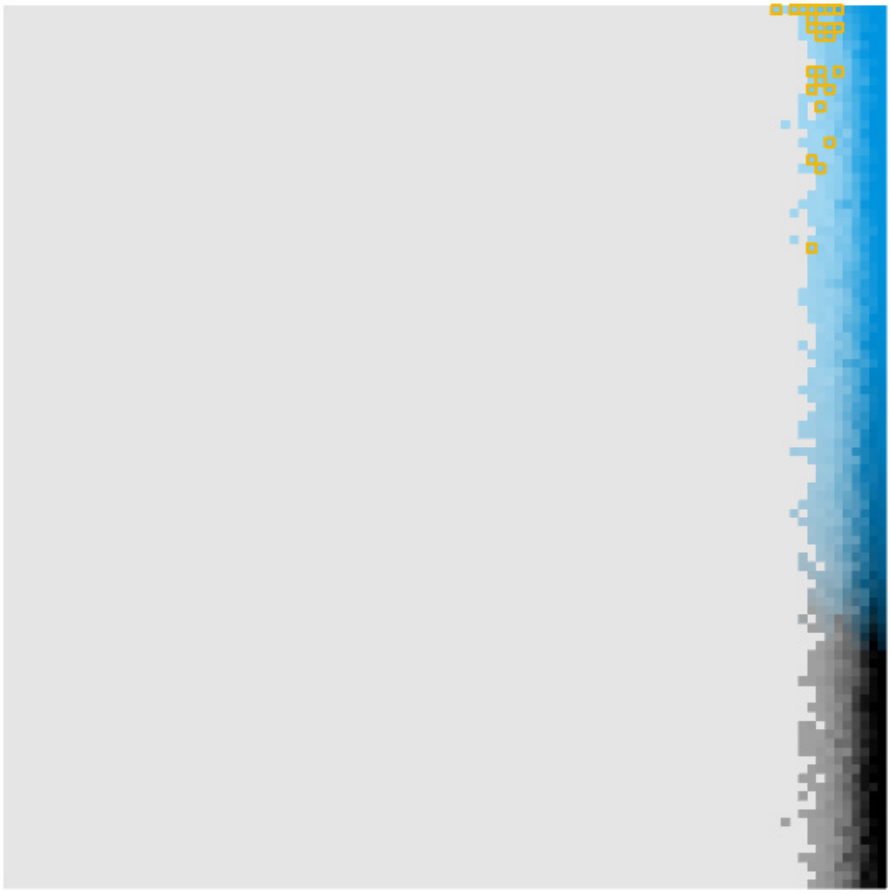


**Fig. 5.23** Eva mode in the natural selection file average. The last column is selected and the result is the highlight of every individual that met the eva mode conditions across every file.

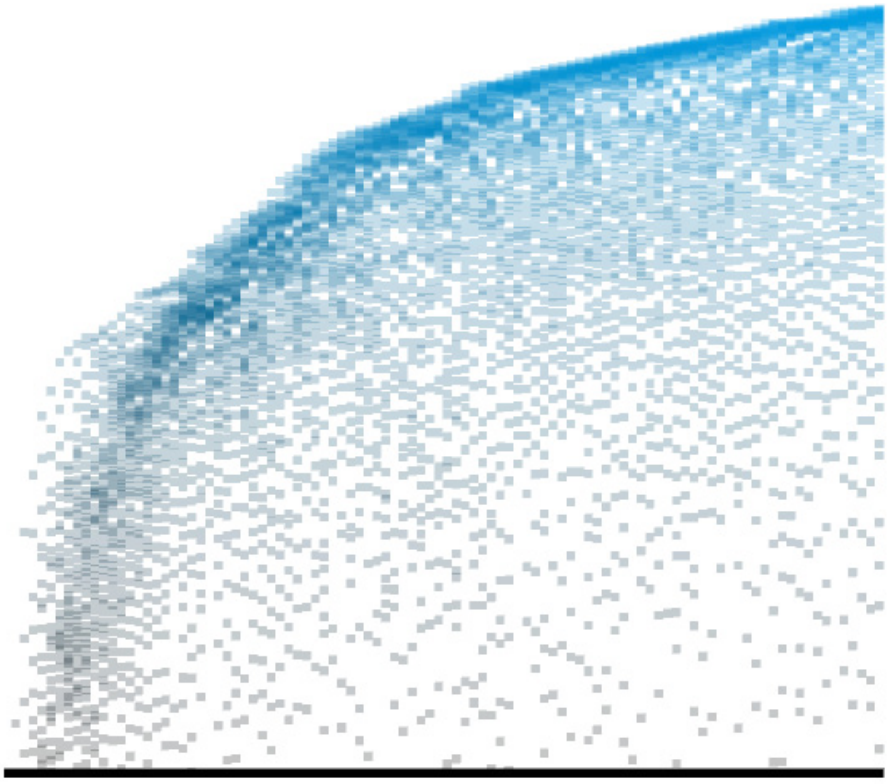


**Fig. 5.24** Position by fitness mode in the natural selection file average. The position mapping works like before but with the average fitness value.

# Individual view

*Individual view* is a different but parallel visualization which focuses on representing all the information of a single individual at a time[4] (Fig. 5.25). As described previously, each individual represents a possible solution to the problem trying to be solved, which in the case of the data being visualized is a search for a function which creates a curve as close to a target curve as possible. This means that each individual contains a function (or two in case of sexual selection) which can be represented as a curve alongside the target curve so that the user can actually visualize how the two curves become more similar as the solutions become more fit. The function's structure contained in the individual is actually a tree, in which the nodes are composed of either variables or mathematical operators and each path of the branches represents a segment of the function. This allows us to represent the function visually as a tree instead of a string of text, which makes it both more visually interesting and easier to discern the differences between various individuals. It is a much more complex process, but necessary because this data is likely to get rather complex after many recombinations, and a graphical representation makes it much easier to understand when compared to rather large written formulas.

One of the main objectives of this visualization is to observe the process that shapes the individuals genes into fitter solutions over time, which can be complicated when any one individual can have a large amount of both ancestors and descendants. The solution was to focus on a single path of connected individuals, selecting only one per generation. This path works as a timeline, where one can move forwards and backwards through the generations and focus on one selected individual for each generation and observe how the genes change in that particular family line.

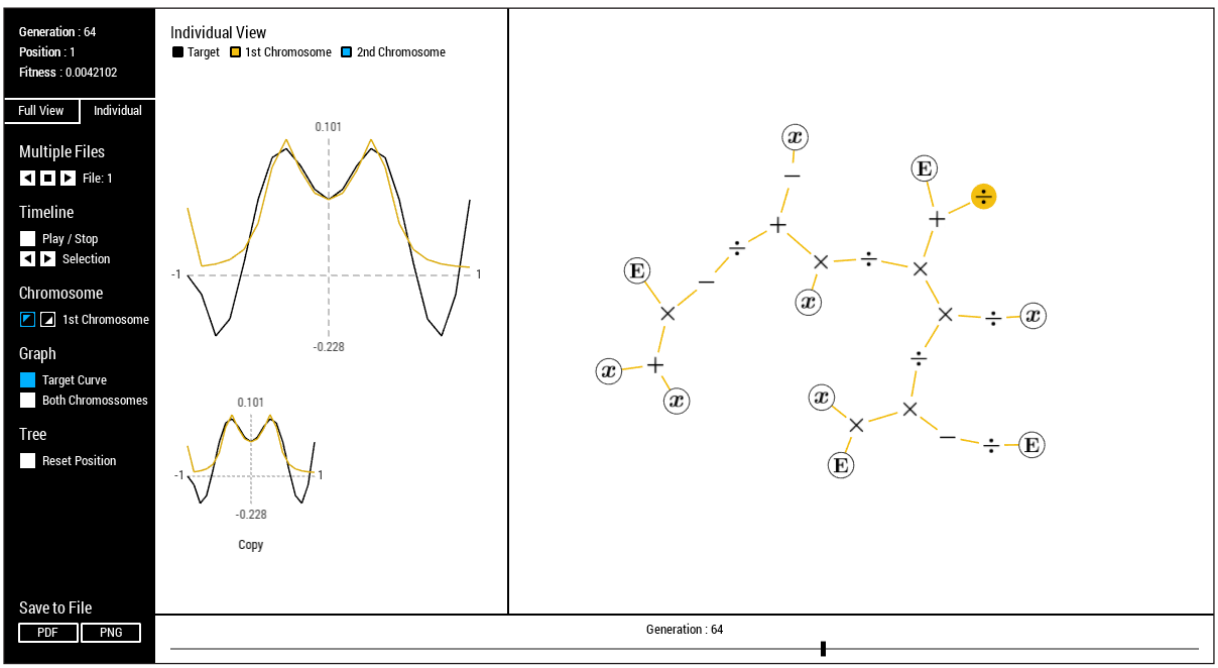



**Fig. 5.25** Individual view consists of a different set of structures used to visualize each individual's data, namely a graph and a tree. The interface also contains a distinct set of functions.



[6] Individual view video: https://vimeo.com/105060415

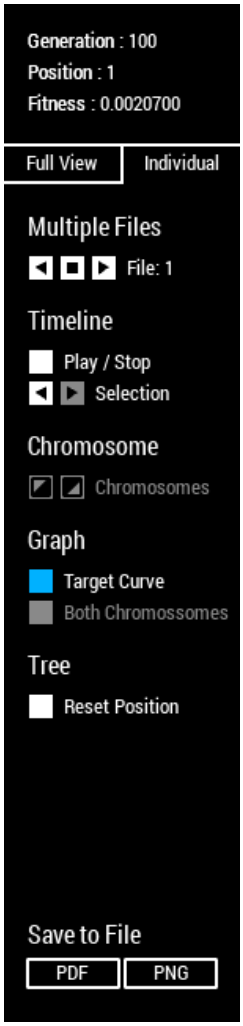



**Fig. 5.26** A close-up on the individual view's interface. You can switch between full and individual view using the tab buttons on the top. Like the full view interface, buttons in grey are not usable, such as those relating to changes in chromosomes when visualizing natural selection data, which only contains information on one chromosome.

When an individual is chosen a timeline is created, and this is achieved by selecting which of the individual's parents and children will be added to the timeline through established rules, which are then repeated and applied to the chosen parents and children until the timeline is completed. The rule for picking the parent is simple: the parent with the highest fitness will be chosen, and if both parents have the same fitness then the mother will be chosen.
A simple method was also used initially to pick the children, where the child with the highest fitness was chosen, but this path would very often lead to a child with low fitness and no descendants and prematurely end the timeline.
A different approach was taken to fix this: when an individual is picked, the best individual of the newest generation which belongs to the selected individual's family line is selected, and then a path of the fittest individuals that connect those two is selected and all the individuals in that path are added to the timeline, assuring that the timeline goes as far as possible. The bottom scroll bar in the individual view is used to navigate through the timeline, allowing the user to click and drag the indicator to the generation he wishes to visualize. The controls on the interface (Fig. 5.26) also allow for some control, such as moving forward or backwards one generation at a time, as well as a "play" button, which initiates an automatic forward sequence on a timer, like an animation which shows the story of the selected individual's family line.

The *view timeline* function is actually in Full View, but it is described in this section because it was necessary to have a basic understanding of what the timeline was, and because it is used mainly as a navigation aid for the individual view. By selecting this option it is possible to visualize which individuals are selected into the timeline as they will be highlighted with a surrounding yellow box as if they were selected (Fig. 5.27). The currently selected individual with be changed to a filled yellow box, to further distinguish it from the rest, much like when visualizing a different individual in the timeline than the currently selected one (Fig. 5.28). Furthermore, if the user opens the graph to visualize the connections, all the connections that aren't part of the timeline will be faded out, while the current timeline will appear as fully opaque.When individual view is selected there must be a timeline to visualize, so there is a default selection which is automatically made if there isn't any individual selected, which is the best individual of the final generation. This was done so that the default selection shows the timeline which lead to the best possible individual. It is also not possible to create a timeline of an entire population, meaning that when a population is selected and the user enters individual view the best individual of that population will be selected for the creation of the timeline. Finally, there is also a small visual indicator which appears if the user was viewing a different individual in individual view and then switched to Full View: the currently selected individual will maintain its surrounding yellow box but the individual the user was viewing will be represented with a filled yellow box. This is because the individual being viewed isn't selected, and instead is part of the selected individual's timeline (Fig. 5.28).

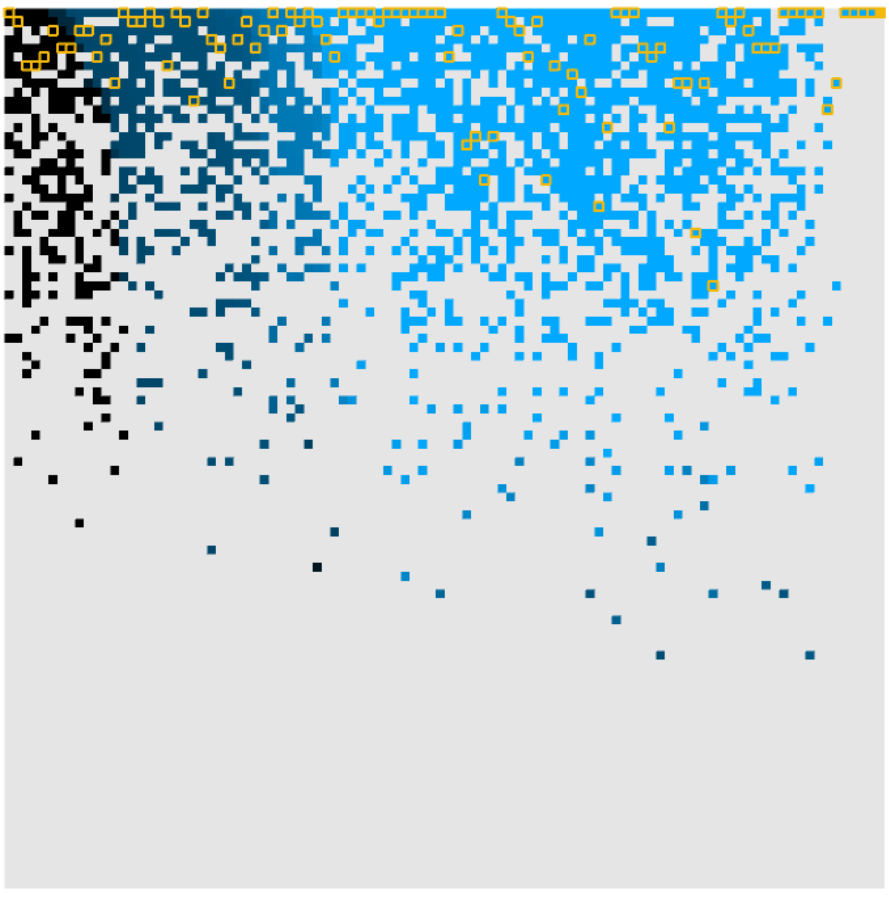


**Fig. 5.27** Visualizing the timeline in full view using the view timeline function. The individual in each generation which belongs to the timeline appears highlighted. This timeline originates from the individual in the top right.

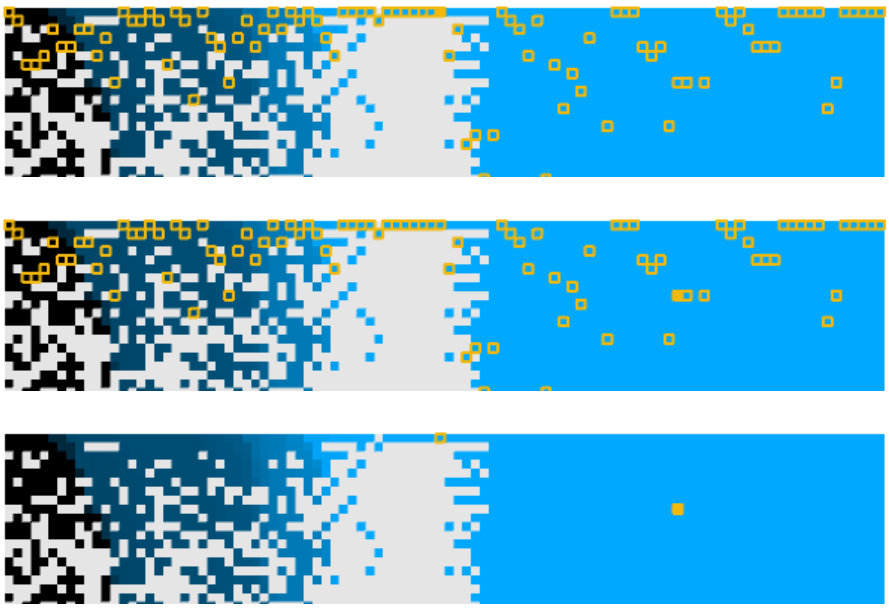


**Fig. 5.28** The top image shows the time line selected for the top middle individual. In the middle image, the timeline remains the same, but the individual selected in the individual view is on the middle left. This is more easily verified on the bottom image, when the view timeline option is off.

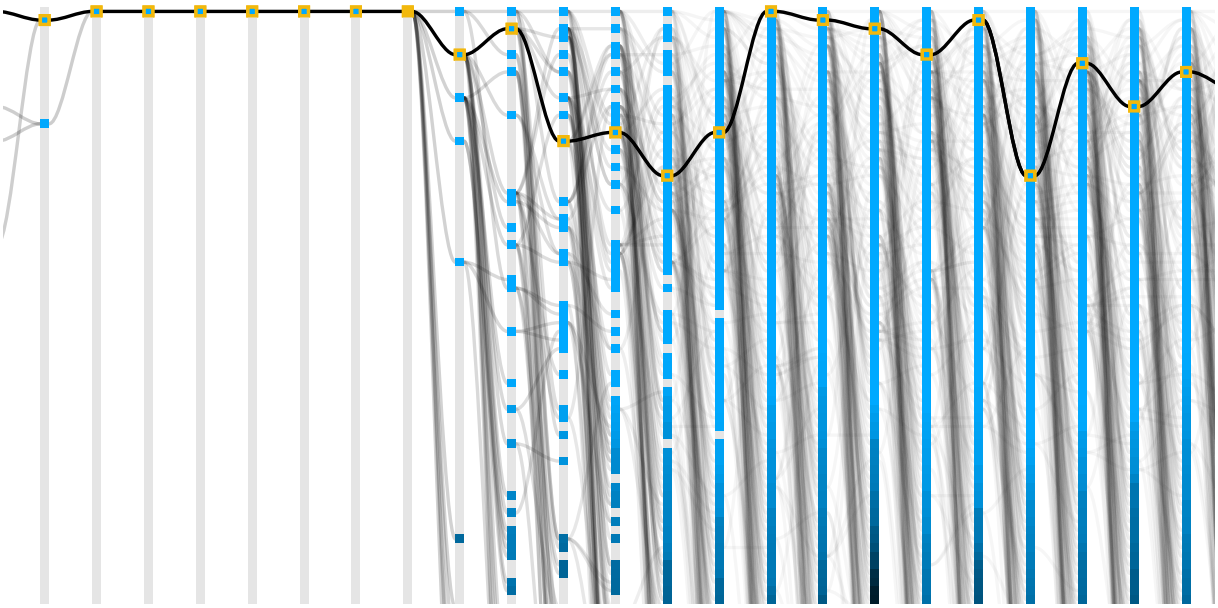


**Fig. 5.28** Open graph visualization whil the view timeline mode is on. All connections become more transparent with the exception of the connection shared between each individual in the timeline.

Regarding the individual's data itself, it is represented through the use of two structures: a graph and a tree. The graph is represented with a very basic style where a line is drawn on a two-dimensional plane representing every value gone through the function (the genotype), within a certain range (Fig. 5.29). This main graph line is colored yellow (like the selection indicator in Full View) and it is built through a dynamic function which adjusts its size to the maximum values so that the whole graph is always represented. However, this could have originated confusion as to the exact scale of the graph as the

maximum values change along with the scale, but this was solved by adding some points of reference, represented with black lines. The main reference point is the target curve which is drawn behind the individual's graph so it can be easily compared. The remaining reference points are two thin dashed lines which represent the x and y axis, along with numerical values which indicates their upper and lower limits. When visualizing the sexual selection data, the user can view both chromosomes' curves at once as they are distinguishable through color, where the currently first chromosome keeps the yellow line, and the second chromosome is represented with a blue line. The user also has the option to switch on and off which curves he wants to see by using the appropriate buttons on the interface, namely the target curve and the curve of the currently unselected chromosome (Fig. 5.30). Drawn below the individual's graph are its parents graphs, as well as a textual indicator of the acting genetic operators, which serve as a quick overview of the origin of the current graph's genotype (Fig. 5.29). During sexual selection, this textual indicator will distinguish the two graphs as "Mother" and "Father", but when displaying natural selection data those graphs will only refer to the operator as "Crossover" because there is no clear distinction between parents, much like the connections' colors in the open graph view described previously.

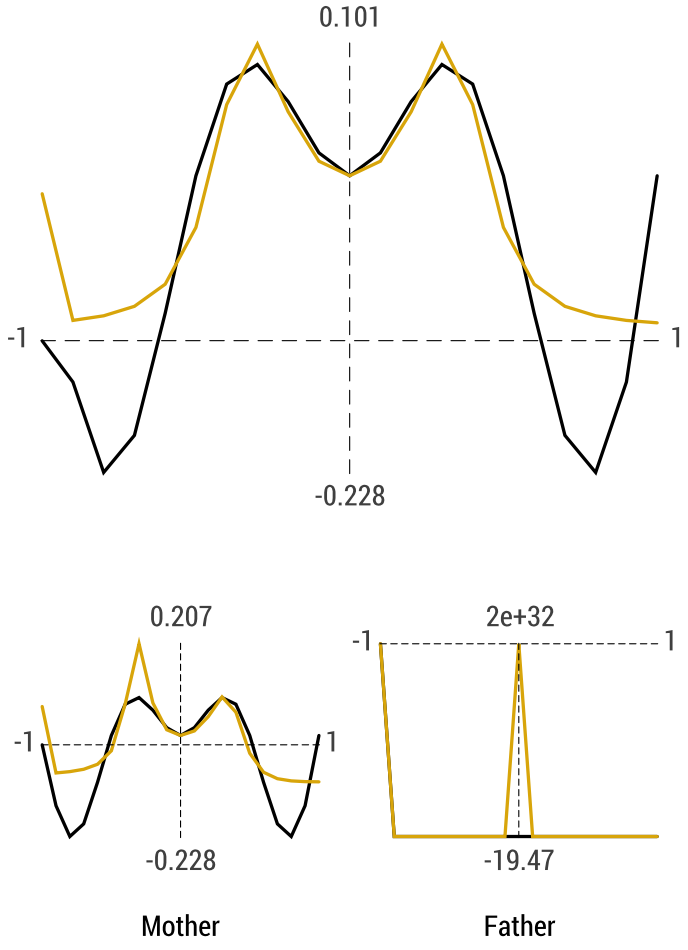



**Fig. 5.29** An example of a graph, shown along with its progenitors, picked from sexual selection data. It shows a crossover between two individuals which results in an individual with a curve which is closer to the target curve.

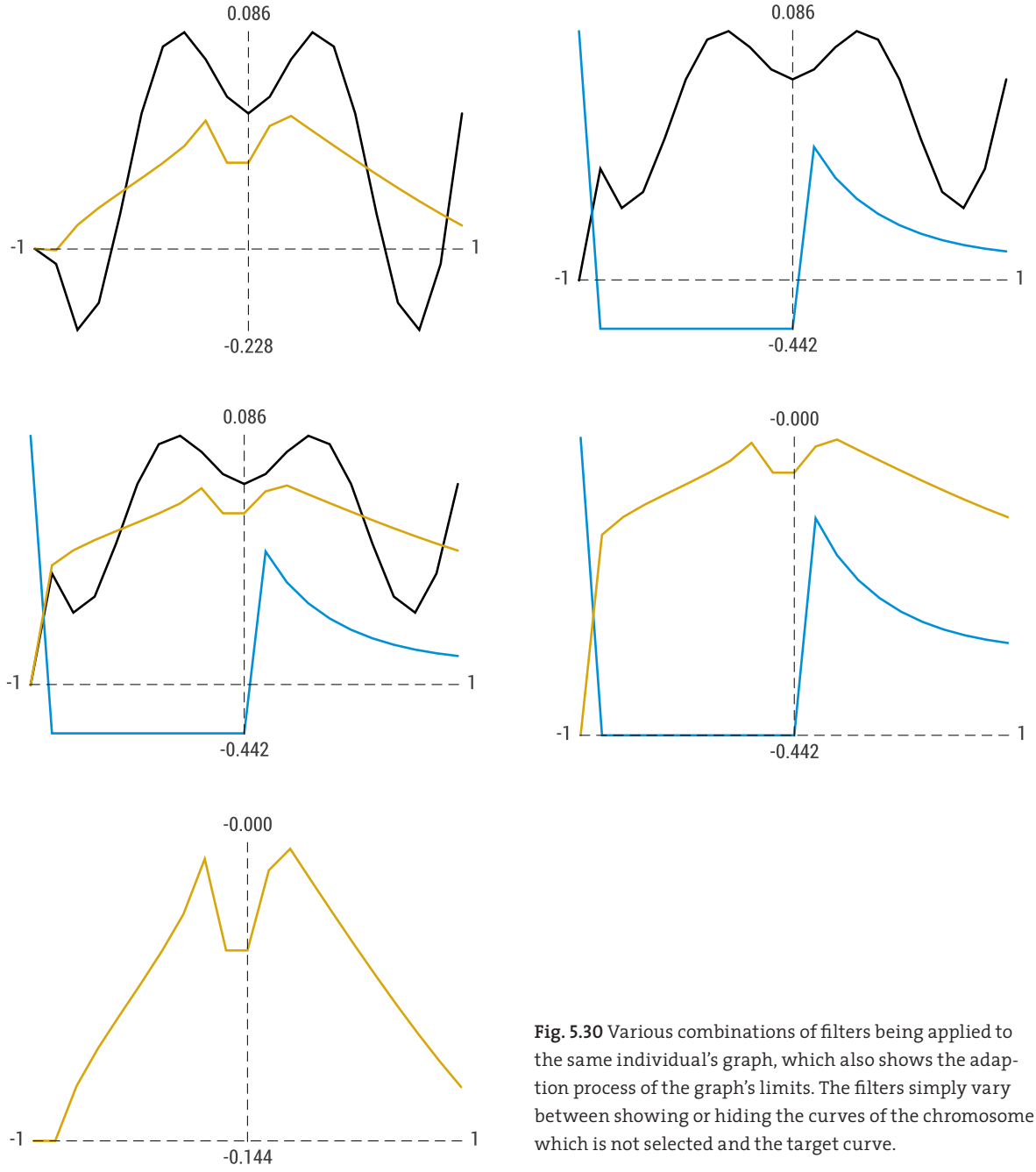



**Fig. 5.30** Various combinations of filters being applied to the same individual's graph, which also shows the adaptation process of the graph's limits. The filters simply vary between showing or hiding the curves of the chromosome which is not selected and the target curve.

The tree can be a much more complex structure to represent as it can have thousand of nodes and branches, and as the program needs to load these structures from the data files without knowing exactly how large they would be, certain precautions had to be taken to guarantee that the user could visualize them both in their entirety and in detail within the limited space. The solution was to implement something similar to the navigation in the Full View visualization, where the user can drag the tree around and zoom in and out, utilizing the exact same controls, meaning that the size and complexity of the tree would no longer be a problem in terms of adapting to the space available. The tree structure itself also had to be designed with a couple of concepts in mind: it had to adapt itself to any size or complexity of data in order to maintain legibility, which means each of the branch paths should be identifiable with as little overlapping branches as possible, and there should also be some discernible elements that would allow the user to identify what mathematical operator corresponded to each node.

In order to tackle the adaptability of the structure to the data we turned to some of the nature-inspired techniques, which are most notable for their ability to sort out large amounts of elements by creating simple interaction rules between them. The organization of a tree structure could also be achieved through a force-based system [26] with simple rules, mainly by having all the nodes in the tree repel each other, along with an attraction force between any two nodes that are connected, assuring that every connected node will maintain a set distance between each other and try to not interfere with any other nodes which aren't directly related. Through a careful calibration of each of the forces values we were able to achieve a balance between them and obtain some favorable results even on some of the more complex trees, though there was still some notable overlapping in some cases. In order to both speed up the organization process and to help prevent overlapping branches the position of the nodes was no longer randomized at the start, and instead a function was built which would calculate a fixed starting position based on what level of the tree they were on (vertical position) and with how many other nodes (horizontal position) (Fig. 5.31). The combination of this function with the force based layout lead to a very quick and effective organization of the nodes which also resulted in the creation of structures that looked more organic than those which would have only used a static positioning function. This process is activated when a new tree is loaded, at which point the user can watch as the tree unravels itself (Fig. 5.32).

Initially the tree would reset its position every time that a different individual was changed, which made for very abrupt transitions, so this was changed so that only the branches which had changed between individuals would have their positions reset. The result was that as the user scrolls through the timeline and visualizes the tree change, the unchanged branches will keep their positions and movement, while the branches that change or appear will reposition themselves and cause their surrounding section to reorganize itself (Fig. 5.33), acting as a visual indicator of the changes that took place without having to resort to style changes, such as different colors.

Our second main concern was the representation of the data value attached to each node on the tree. Some of the initial ideas considered different colored nodes with textual subtitles or just small textual indicators next to the nodes, but due to the data density of some of the trees an approach which was more simple and direct was necessary. Each of the nodes can have only one of seven values associated with it, which consist of mathematical operators and variables: a sum, subtraction, multiplication, division, square root, a variable representing the x axis, and an ephemeral random constant (ERC). Given that we were working with this limited range of values, we decided to use simple and easily identifiable symbols which represented each of them accordingly, since most of them already had a widely known mathematical symbol, namely the sum (+), subtraction (–), multiplication (x), division (÷) and the square root (√). Given the density of some of the trees we took a simplistic approach to their design: the symbols appear on the position of the nodes and are connected with thin lines and only the leaf nodes are circled with a thin stroke in order to distinguish themselves as the end nodes. The root node is colored in order to be the most notable since it is at the top of the hierarchy, and its color matches the color of its respective line on the graph, meaning that it will be yellow when representing the first chromossome's tree and blue when representing the second (Fig. 5.34).

The symbols are written using the bold weight of the *Euclid* typeface, a modern typeface family designed by the *Design Science* group, used in their software *MathType*, a programs for editing mathematical equations [30]. The choice was based on how well the font could preform the task of representing all of the mathematical symbols that were contained in the data in such a way that they were easily recognizable, which the current main font, Roboto, could not do on its own. Since the variable is supposed to represent the x axis of the graph, its symbol should be an x, however this cannot be confused with the multiplication sign, which is why an italic, cursive x was picked, making it very distinct and easily identifiable. The ERC was simply represented with a capital E, however an extra function was added to this particular symbol do the fact it has got a constant value associated with it, allowing the user to visualize the value next to the symbol when the user hovers the mouse on it. In order to avoid overlapping, the text with the value will appear on the side of the symbol which is opposite to the center of the graph, which is more likely to have less branches (Fig. 5.35). Every symbol was written using the Euclid Symbol variant of the family, with the exception of the x, which was written with the regular variant of the typeface.

Aside from the described navigational functions there is also a function located on the interdace which allows some more control over the tree representation called "reset position". This is a simple function which resets the position of both the tree itself (by centering it) and all of its nodes. It was added because the nature-inspired method which organizes the tree's nodes isn't infallible, particularly when adding or changing branches which can result in some undesired overlapping. However this isn't a common occurrence, and the function also exists in case the user simply wants to see the animation of the tree organizing itself from the start.

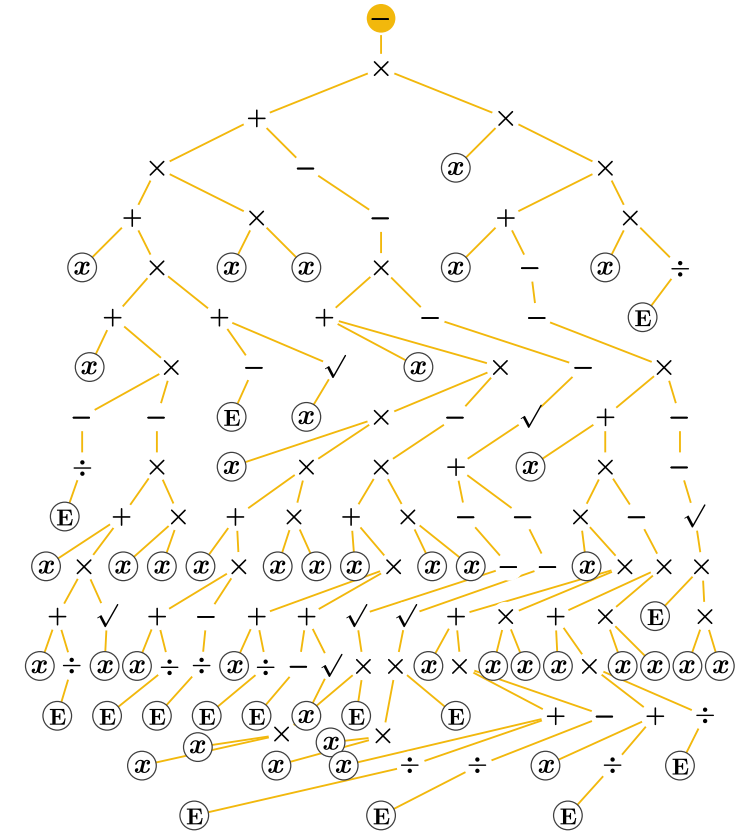


**Fig. 5.31** If the tree is being loaded for the first time or if its positions are reset, the branches will be positioned below the root node in accordance to their level on the tree and the number of nodes on that level.

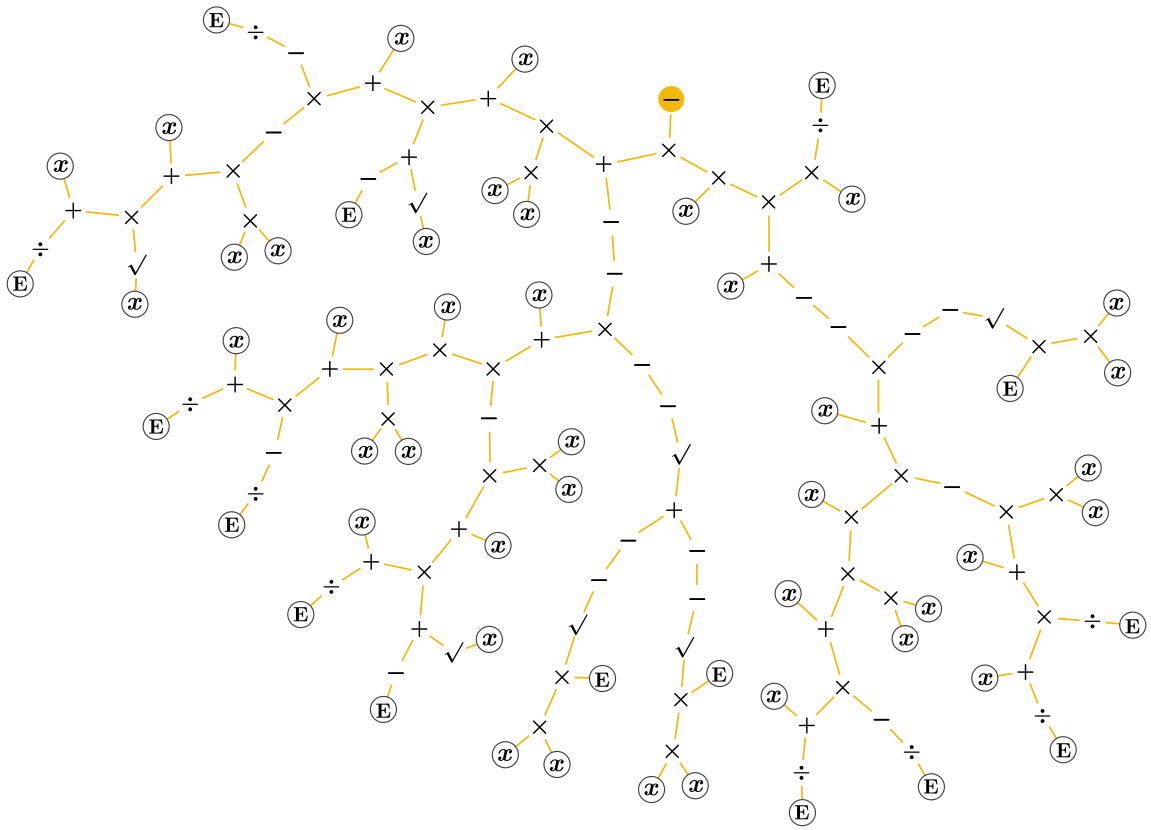


**Fig. 5.32** The tree represented in this image is the same as the one in Fig. 5.31 after a few seconds have passed, as the nodes reposition themselves due to the force based layout.

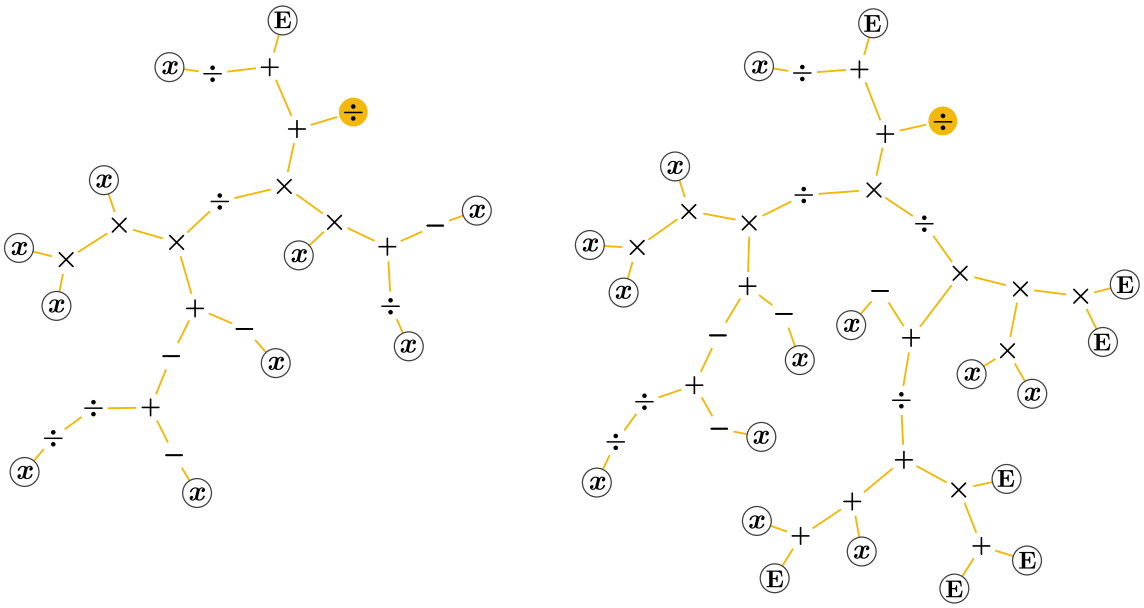


**Fig. 5.33** As the user navigates through the timeline the tree's branches represent any existing changes and reposition themselves in order to adapt to these changes. This image shows as a new branch is added to the tree on the left, resulting in the tree on the right.

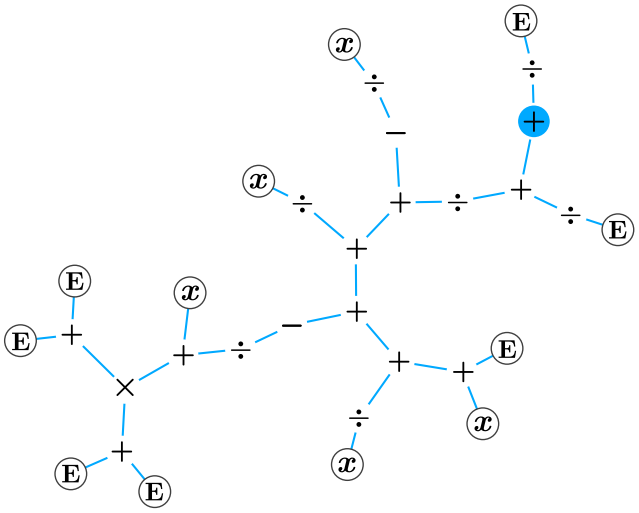


**Fig. 5.34** When the tree represents the second chromosome of an individual, its node will be blue to match its respective graph's curve.

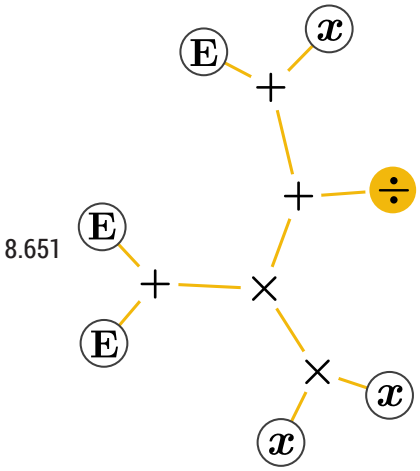



**Fig. 5.35** When the mouse hovers the ephemeral random constant (E) symbol there will be an indicator which shows the value of the constant in that node.

When visualizing a file average, all of the individuals located in the same position (row and line) are merged in order to show an aggregate of all their data so that one can find a distinguishable pattern when comparing these when overlaid. This was applied to the graphs by overlaying the curves of the functions of each individual with some transparency, which depends on the amount of individuals being displayed (more curves requires a higher transparency value), while allowing the target curve to still be present in the background so that they may be compared to it as well. However, the tree is a much more complex structure which cannot simply be overlaid in order to draw the same conclusions since the variation in size and branches would cause a lot of confusion, especially when dealing with a lot individuals at once. Because of this the tree structure is not present in the individual view with the file average option selected. Another function which did not work across multiple files was the ability to build and navigate through a timeline because there are no consistent relationships between individuals across multiple files and a timeline could not be properly selected using averages, unless we risked selecting individuals which were not connected which would be inaccurate to the process.

Regarding the “save to fle” function during individual view, as there are more types information being displayed on the screen, saving to file will result in a simple screenshot which will save the graph and tree as they are being shown at that moment, but without the interface and horizontal scroll bar.

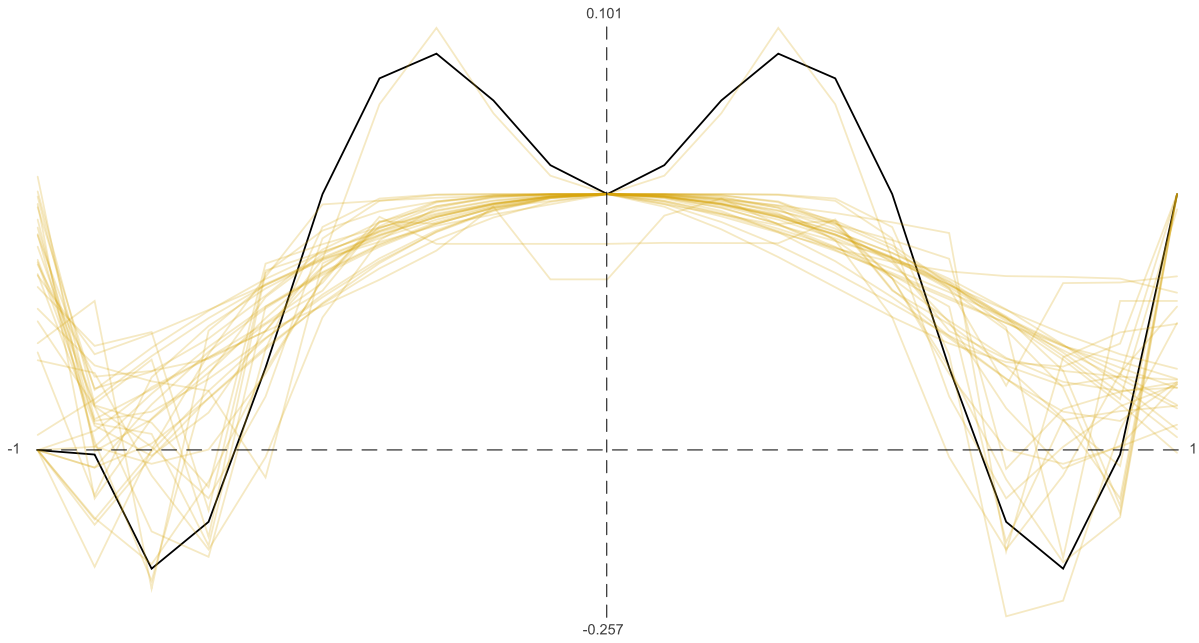



**Fig. 5.29** Graph showing the file average of the thirty best individuals of the natural selection data.

# Results overview

The visualizations we managed to create from the data allowed us to make a much more deeper analysis than through more conventional methods, which usually rely more on looking only at the best overall results. However, the scope of this dissertation is still focused primarily on information visualization and not artificial intelligence, meaning that our result analysis will mainly consist of identifying and analyzing the patterns of information within the context of genetic algorithms. It also means our analysis will be focused on what we can identify from the data we received, and not on the parameters of the genetic algorithm which originated it, given that the algorithm itself was handled by a different group.

Some of the initial representations already allowed us to visualize how quickly the fitness would increase throughout the populations and just how different these results would vary between each run of the algorithm, especially through the use of the "position by fitness" option. Selection of individuals and populations allowed us to observe the percentage of the population which was picked to pass on its genes to the next one, and in some cases how long it could take for a new individual to surpass the past best, which could be copied through elitism through many generations. Some of the worst runs did not manage to generate better individuals throughout many of its recombinations which resulted in long intervals where the best fitness values would remain throughout most of its generations. If, for example, this had occurred more consistently throughout the runs it could have meant a lack of genetic variation in the data which could have been seen as a indication that the mutation rate should have been increased.

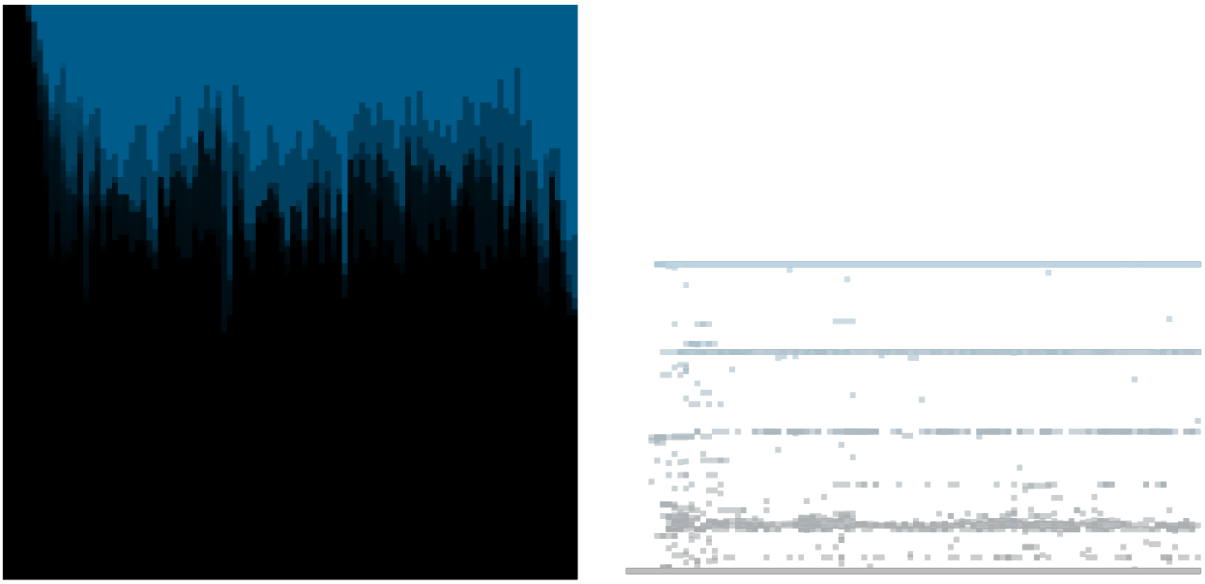


**Fig. 5.29** Fitness view of an individual which was not able to create individuals with better fitness than each previous population after the first few generations. Because of elitism the current best kept being copied from each previous population, which can easily be seen in the position by fitness mode on the right, represented by the top horizontal line.

Eva mode was also very useful in revealing information which would have been impossible to discern with the textual data alone, namely just how quickly a fit individual's genes can spread through an entire generation. Fit individuals can have many children which allows their genes to spread very quickly, which can be better observed in the open graph visualization through the large increase in connections from generation to generation. Looking at the genetic operators we could also see some patterns that indicated that individuals who were picked to be copied tended to be fit, while individuals created from mutations were often unfit, though this is still a necessary process because mutations add genetic variation which is often necessary to generate better solutions rather than constant recombinations of the same genes.

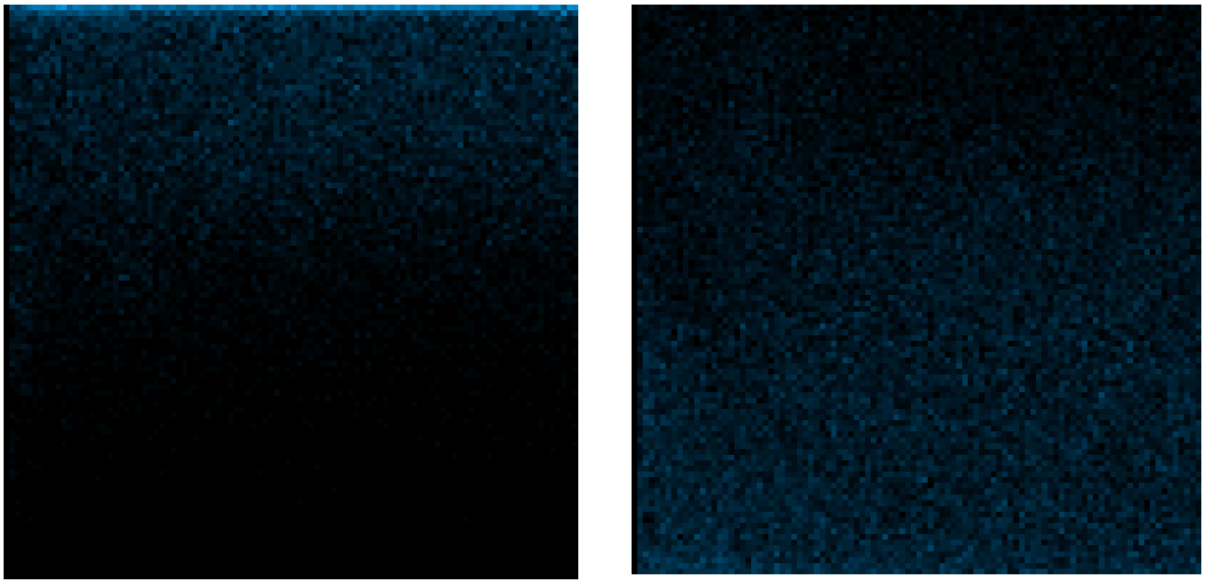


**Fig. 5.29** Average file visualization of the genetic operators of the natural selection data. Copies are represented on the left image, showing a concentration of copies on the top, and mutations are represented on the right image, which shows a larger concentration of mutations of the bottom.

It was also possible to discern large differences between the natural selection data and the sexual selection data on multiple aspects. Regarding their fitness, sexual selection has a larger percentage of individuals with poorer fitness and seems to have less fitness variation in its results. The individuals selected to pass on their genes is also quite different, clearly visible in the children view where most of the natural selection individuals which have children are fit individuals, while the sexual selection individuals chosen seem to be both the most and least fit (Fig. 5.21). Using the open graph visualization in sexual selection we can also clearly visualize that the fit individuals are being picked as mothers while the unfit are picked as fathers for the creation of new individuals (Fig. 5.9).

This analysis of the population's evolution was further helped with the addition of the individual view which allowed us to visualize the changes that take place between each generation in greater detail. Visualizing the individuals evolving is fairly simple and quick by scrolling through the timeline and it is easy to identify at which moments its genes underwent the biggest changes, as well as the conditions which lead to those changes. This can be seen through both the graph, which serves as a direct comparison to the intended solution, and the tree, which allows the user to visualize the complexity of the function which is currently being shown and the intensity of the changes between each generation.

The various functions and modes implemented allow for a large degree of interaction which give the user control what he wants to see and how, which is how we were able to address the main concern of having to represent large quantities of data without having to sacrifice neither legibility nor information, while also adding a new dimension which helps captivate and maintain the user's interest.[7, 8, 9]

[7] Full view video: https://vimeo.com/105063501

[8] File average video: https://vimeo.com/105063501

[9] Individual view video: https://vimeo.com/105063501

# Chapter 6
# Conclusions and future work

Throughout the dissertation we discussed the influence of the field of information visualization in the past and present, as well as how it continues to change and evolve as we move towards the future. As we delved into the creation of a visualization tool it was necessary to study the existing work in the field in order to both use it as a point of reference and to expand upon it as we seek to build new visualization tools. The history of information visualization extends as far as early as the creation of the first maps and tabular presentations and these techniques have since been been polished in order to represent information more efficiently and more effectively, aided with the creation of technological tools which allowed for the processing of extremely large datasets automatically and represent them graphically in such a way that the viewer can easily understand.

The techniques we still use today are still based on some of the most important early developments in the field, such as Jaques Bertin's theories which documented some of the most essential principles of graphical communciation and established the basic rules for the representation of information, and Edward Tufte's work which pointed out some of the biggest problems surrounding information visualization and how they could be corrected or avoided. These were some of the basic rules and concepts that we covered, but these approached data visualization in a rather broad way, and this is because they act as effective standards to have during the development of most visualizations. However, in order to properly represent the data, the visualization needs to be built with it in mind, meaning that different datasets can entail completely distinct visualization structures.

During the first semester we dedicated ourselves to planning and research but there were some limitations due to the dataset being unavailable, making it necessary to consider the representation of different types of visualization structures in order to prepare for the representation of the data which we would be able to use. A more practical approach was taken with the development of the project for the Massachusetts Institute of Technology's contest, and while it was not directly linked to our own project, it acted as a preparation for the creation of the application at the center of this dissertation. As we entered the second semester we were able to obtain the desired dataset and refine and focus our research and development. This data contained all the

information generated and processed by various runs of a genetic algorithm, a search heuristic used for finding and optimizing solutions to particular problems by using methods based on natural selection. After generating an initial population of individuals, new populations are the progressively created through the recombination and mutation of a selection of the best individuals from each previous generation. Our objective was to represent theses generations and their changes with each iteration, allowing the observer to identify and analyze any of the individuals generated and both their history from past generations and influence over the following generations.

We were able to build a functional prototype capable of loading and processing the external data files we had received and then display them on screen by using the appropriate visualization structures. The main network structure depicted the proprieties of each individual, such as their fitness and genetic operators, and their relationships with each other, achieved by establishing connections with their progenitors and descendants. The visualization of the data was aided by the integration of interactive functions which provided various options to the user and gave him control over the information he was viewing, allowing him to view specific parts of the visualization in more detail, filter and switch between different information, navigate through the data using a variety of methods, and export the current visualization into a static artifact. These interactions were programmed to be achieved through simple commands using the mouse and the interface.

The main visualization structure that we focused on building initially was the network, due to the type of data and relationships which would have to be represented, although we also turned to tree structures in order to represent a particular set of data that related to the more detailed information of each individual. These were structures that we also approached in the dissertation in order to have a more in-depth knowledge on their proprieties so that we could build our own. Temporal structures were also studied, and while the data was not temporal in the usual sense (using defined dates or timestamps), the populations of individuals were simulating various sequential generations and we were able to adapt useful concepts from temporal structures into the navigation and certain other functions, such as the creation of timelines.
The application allowed us to produce various visualizations of the data we worked with, but one of our main objectives was to do more than represent information on screen. The functions implemented served their purpose in aiding the user navigate through all the information available and observe it through various points of view or levels of detail, and allowed the user to quickly search through the information, analyze the intricate relationships between the data, and identify significant patterns which could lead to meaningful conclusions.

Most of the future work regarding this project would be in the interest of turning this application into a proper tool which could be used by anyone which required its services. The application was created with certain limitations in mind, the biggest one being the limited time available for its development. This meant that our focus was on representing the datasets we had available, and so the program is structured to deal with a rather specific

data file structure, which can be seen as a limitation. By obtaining more data we could start to develop better error prevention methodologies which would allow the user to easily figure out what files he could use with the application or to indentify the problems with is current files more promptly. A more concrete example of unprepared exceptions are the individual representations for which there are only seven symbols that can represent the data values used in their trees, since these were the only ones present in our data files, but these could be expanded upon in order to be prepared to recieve any other kind of data value and still represent it effectively. More advanced techniques to receive feedback could also be employed such as focus groups which would allow for futher user testing, allowing us to create an interface which would always assure intuitive interaction with the average user.

In parallel with the work developed in the context of this dissertation we also explored the use of swarm based approaches for the visualization of music. This bio-inspired approach was the winner of the Science Visualization Competition, Visual Media Category of the *ALIFE 14: The Fourteenth International conference on the synthesis and Simulation of Living Systems*, July 30th – August 2nd, 2014, New York, NY, USA.

# References


[1] J. Bertin, *Semiology of Graphics : Diagrams, Networks, Maps*, 1st ed. Redlands, California: ESRI Press : Distributed by Ingram Publisher Services, 2010.

[2] E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Cheshire, Connecticut: Graphics Press, 2001.

[3] M. Friendly, *A Brief History of Data Visualization*, Handbook of Computational Statistics: Data Visualization, vol. III, Springer-Verlag, Heidelberg, 2006.

[4] E. R. Tufte, *Beautiful Evidence*, 1st ed. Cheshire, Connecticut: Graphics Press, 2006.

[5] J. Tukey, *The Future of Data Analysis*, The Annals of Mathematical Statistics, vol.33, no.1, pp.1-67, 1962.

[6] I. Meirelles, *Design for Information*, Beverly, Massachussets: Rockport Publishers, 2013.

[7] A. MacEachren, *How Maps Work: Visualization*, Representation and Design, New York, The Guilidford Press, 1995.

[8] M. Lima, *Visual Complexity* : Mapping Patterns of Information, 1st ed. New York, New York: Princeton Architectural Press, 2011.

[9] M. Lima, *The Book of Trees: Visualizing Branches of Knowledge*, Princeton Architectural Press, 2014.

[10] G. Judelman, *Aesthetics and inspiration for visualization design: Bridging the Gap Between Art and Science*, Proceedings of the Eighth International Conference on Information Visualisation, pp.5, 2004.

[11] A. Moere, *A Model for Self-Organizing Data Visualization Using Decentralized Multiagent Systems*, Advances in Applied Self-organizing Systems, Springer-Verlag London Limited, Part III, pp.291-324, 2008.

[12] C. Reynolds, *Flocks, Herds, and Schools: A Distributed Behavioral Model*, Computer Graphics, SIGGRAPH '87 Conference Proceedings, pp.25–34, 1987

[13] R. Beale, R. Hendley, A. Pryke, B. Wilkins, *Nature-inspired visualisation of similarity and relationships in human systems and behaviours*, Information Visualization, vol.5, no.4, pp.260-270, 2006.

[14] C. Darwin, *On the Origin of Species*, John Murray, London, Albemarle Street, 1859.

[15] C. Darwin, *The Descent of Man and Selection in Relation to Sex,* John Murray, London, Albemarle Street, 1906.

[16] J. Holland, *Adaptation in Natural and Artificial Systems,* The MIT Press, Cambridge, Massachusetts, London, England, fifth printing, 1998.

[17] M. Mitchell, *An Introduction to Genetic Algorithms,* The MIT Press, Cambridge, Massachusetts, London, England, fifth printing, 1999.

[18] G. Naik, *Back to Darwin: In sunlight and cells, science seeks answers to high-tech puzzles*, The Wall Street Journal, January 16, pp.A1, 1996.

[19] E. Altshuler, D. Linden, *Design of a wire antenna using a genetic algorithm, Journal of Electronic Defense*, vol.20, no.7, pp.50-52, 1997.

[20] S. Obayashi, D. Sasaki, Y. Takeguchi, N. Hirose, *Multiobjective evolutionary computation for supersonic wing-shape optimization*, IEEE Transactions on Evolutionary Computation, vol.4, no.2, pp.182-187, 2000.

[21] A. Leitão, J. Neves, P. Machado, *A self-adaptive mate choice model for symbolic regression*, IEEE Congress on Evolutionary Computation, pp.8-15, 2013.

[22] A. Leitão, P. Machado, *Self-adaptive mate choice for cluster geometry optimization,* Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013, pp.957-964, 2013.

[23] W. Banzhaf, P. Nordin, R. Keller, F. Francone, *Genetic Programming - An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*, Morgan Kaufmann Publishers, San Francisco, California, 1998.

[24] B. Fry, *Visualizing Data*, 1st ed. Beijing; Cambridge: O'Reilly Media, Inc., 2008.

[25] N. Elmqvist, A. Moere, H. Jetter, D. Cernea, H. Reiterer, T. Jankun-Kelly, *Fluid Interaction for Information Visualization*, Information Visualization, vol. 10, pp. 327-340, West Lafayette, Indiana, 2011.

[26] S. Kobourov, *Spring Embedders and Force Directed Graph Drawing Algorithms*, Cornell University Library, arXiv:1201.3011v1, 2012.

[27] J. Holland, *Hidden Order: How Adaptation Builds Complexity*, Helix Books, Addison-Wesley Publishing Company, pp.11, 1995.

[28] C. Robertson, *Roboto*, Google Fonts, 2011 [Online], available: www.google.com/fonts/specimen/Roboto [Accessed: 1 September 2014].

[29] P. Casteljau, *Polar Forms for Curve and Surface Modeling as Used at Citroen*, Fundamental Developments of Computer-Aided Geometric Modeling, Les Piegl, Academic Press, pp.1-12, 1993.

[30] Design Science, *MathType 6.9*, 2013, [Online], available: http://www.dessci.com/en/products/mathtype/ [Accessed: 2 September 2014].

Fig. 2.1 Unknown author, *Heavenly Bodies*, in: M. Friendly, *A Brief History of Data Visualization*, Handbook of Computational Statistics: Data Visualization, vol. III, Springer-Verlag, Heidelberg, p.4, 2006.

Fig. 2.2 C. Scheiner, *Sunspots*, 1626, in: M. Friendly, *A Brief History of Data Visualization*, Handbook of Computational Statistics: Data Visualization, vol. III, Springer-Verlag, Heidelberg, p.5, 2006.

Fig. 2.3 J. Lambert, *Pyrometrie*, 1779, in: E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Cheshire, Connecticut: Graphics Press, p.29, 2001.

Fig. 2.4 W. Playfair, *Exports and Imports to and from Denmark & Norway from 1700 to 1780*, 1786, in: *Wikipedia Commons*, 2011 [Online], available: commons.wikimedia.org/wiki/File:Playfair_TimeSeries-2.png [Accessed: 27 January 2014].

Fig. 2.5 W. Playfair, *Chart, Shewing at One View The Price of The Quarter of Wheat, & Wages of Labour by the Week, from The Year 1565 to 1821*, 1821, in: E. R. Tufte, *The Visual Display of Quantitative information*, 2nd ed. Cheshire, Connecticut: Graphics Press, p.34, 2001.

Fig. 2.6 A. Guerry, *Donations aux pauvres*, 1833, in: *Princeton University Library - Sociology and Economics* [Online], available: libweb5.princeton.edu/visual_materials/maps/websites/thematic-maps/quantitative/sociology-economics/sociology-economics.html [Accessed: 27 January 2014].

Fig. 2.7 L. Perozzo, *Numero Absoluto dei Nati Vivi Maschi e loro superstiti classificati per eta secondo i risultati dei Censimenti in Svezia 1750-1875*, 1880, in: *Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualization 1850-1899* [Online], available: euclid.psych.yorku.ca/SCS/Gallery/milestone/sec6.html [Accessed: 27 January 2014].

Fig. 2.8 J. Snow, *Cholera Deaths in Central London*, 1854, in: *Michael Sandberg's Data Visualization Blog* [Online], available: datavizblog.com [Accessed: 27 January 2014].

Fig. 2.9 F. Nightingale, *Diagram of the Causes of Mortality in the Army in the East*, 1857, in: *Wikipedia Commons*, 2010 [Online], available: commons.wikimedia.org/wiki/File:Nightingale-mortality.jpg [Accessed: 27 January 2014].

Fig. 2.10 C. Minard, *Figurative Map of the successive losses in men of the French Army in the Russian Campaign 1812-1813*, 1869, in: E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Cheshire, Connecticut: Graphics Press, 2001.

Fig. 2.11 H. Beck, *London Underground Map*, 1931, in: *Sian Jamieson - Marketing & Development in the Highlands*, 2013 [Online], available: sianjamieson.wordpress.com [Accessed: 27 January 2014].

Fig. 2.12 H. Chernoff, *Chernoff's faces*, 1973, in: *The Use of Faces to Represent Points in K-Dimensional Space Graphically*, Herman Chernoff, Journal of the American Statistical Association, vol. 68, no. 342, p.363, 1973.

Fig. 2.13 E. Turner, L*ife in Los Angeles*, 1977, in: *Catatan Dosen - Just another catatan dosen weblog*, 2008 [Online], available: dosenlog.wordpress.com [Accessed: 27 January 2014].

Fig. 2.14 New York Times, *New York City's Weather for 1980*, 1981, in: E. R. Tufte, *The Visual Display of Quantitative information*, 2nd ed. Cheshire, Connecticut: Graphics Press, p.30, 2001.

Fig. 2.15 J. Thorp, *Just Landed*, 2009, in: *blprnt.blg - Data & Art Miscellanea from Jer Thorp*, 2009 [Online], available: blog.blprnt.com/blog/blprnt/just-landed-processing-twitter-metacarta-hidden-data [Accessed: 27 January 2014].

Fig. 2.16 M. Stefaner, *Visualizing Information Flow in Science*, 2009, in: *well-formed. eigenfactor.org*, 2009 [Online], available: well-formed.eigenfactor.org [Accessed: 27 January 2014].

Fig. 2.17 P. Steinweber, A. Koller, *Similar Diversity*, 2007, in: *Similar Diversity* [Online], available: similardiversity.net/about/ [Accessed: 27 January 2014].

Fig. 2.18 S. Brautigam, *Overnewsed but uninformed*, 2008, in: S. Brautigam, *Overnewsed but Uninformed*, Fachhochschule Mains, University of Applied Science, p.17, 2007.

Fig. 2.19 J. Späth, M. Rembold, *Graphical Visualization of Textual Similarities*, 2006, in: J. Späth, M. Rembold, *munterbund.de,* 2003 [Online] available: www.munterbund.de/visualisierung_textaehnlichkeiten/sketches.php [Accessed: 27 January 2014].

Fig. 2.20 Linkfluence, *Diseasome*, 2009, in: *Diseasome - Explore the human disease network* [Online], available: diseasome.eu [Accessed: 27 January 2014].

Fig. 2.21 A. Adai, E. Marcotte, *Minimum Spanning Protein Homology Tree*, 2002, in: I. Meirelles, Design for Information, Beverly, Massachussets: Rockport Publishers, p.140, 2013.

Fig. 2.22 B. Lyon, *Opte Project*, 2003, in: *—BH Digital Media Production//,* 2012 [Online], available: mynameiskavi.wordpress.com/2012/04/20/data-visualisation/ [Accessed: 27 January 2014].

Fig. 2.23 I. Meirelles, *Bertin's system of perceptual varaibles*, in: I. Meirelles, Design for Information, Beverly, Massachussets: Rockport Publishers, p.127, 2013.

Fig. 2.24 Pravda, *Information chart*, in: E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Cheshire, Connecticut: Graphics Press, p.72, 2001.

Fig. 2.26 P. Baran, *Network Models*, 1964, in: M. Lima, *Visual Complexity : Mapping Patterns of Information*, 1st ed. New York, New York: Princeton Architectural Press, p.55, 2011.

Fig. 2.27 J. Bertin, *Network chart*, in: J. Bertin, *Semiology of Graphics : Diagrams, Networks, Maps*, 1st ed. Redlands, California: ESRI Press : Distributed by Ingram Publisher Services, p.270, 2010.

Fig. 2.28 M. Lima, *Network chart*, in: M. Lima, *Visual Complexity : Mapping Patterns of Information*, 1st ed. New York, New York: Princeton Architectural Press, p.158, 2011.

Fig. 2.29 J. Fekete, *LRI Co-authorship Network*, 2007, in: M. Lima, *Visual Complexity: Mapping Patterns of Information*, 1st ed. New York, New York: Princeton Architectural Press, p.105, 2011.

Fig. 2.30 D. Holten, *Hierarchical Edge Bundles*, 2006, in: *Seeing Complexity - Visualizing Complex Data*, 2011 [Online], available: seeingcomplexity.wordpress.com/2011/02/05/hierarchical-edge-bundles/ [Accessed: 27 January 2014].

Fig. 2.31 P. Cruz, Senseable City Lab, *Data Lens*, 2012, in: P. Cruz, *Visualization et al.*, 2012. [Online]. Available: pmcruz.com/information-visualization/data-lenses [Accessed: 27 January 2014].

Fig. 2.32 M. Porpora, DensityDesign, *The Poverty Red Thread*, 2008, in: datavisualization.ch, *We Love Datavis* 2011 [Online], available: datavis.tumblr.com/post/4416114492/the-poverty-red-thread [Accessed: 27 January 2014].

Fig. 2.33 J. Spahr, *Website Traffic Map*, 2003, in: J. Spahr, *james.spahr.org*, 2003 [Online], available: http://james.spahr.org/193238/2064825/gallery/website-traffic-map [Accessed: 27 January 2014].

Fig. 2.34 W. Paley, *TextArc: Alice in Wonderland*, 2009, in: W. Paley, *TextArc - An alternate way to view a text*, 2009 [Online] available: www.textarc.org/Alice.html [Accessed: 27 January 2014].

Fig. 2.35 E. Haeckel, *Monophyletic Family Tree of Organisms*, 1866, in: : I. Meirelles, *Design for Information*, Beverly, Massachussets: Rockport Publishers, p.26, 2013.

Fig. 2.36 S. Posavec, G. McInerny, *(En)tangled Word Bank*, 2009, in: : I. Meirelles, *Design for Information*, Beverly, Massachussets: Rockport Publishers, p.16, 2013.

Fig. 2.37 M. Weskamp, D. Albritton, *Newsmap*, 2004, in: M. Weskamp, *newsmap*, 2004 [Online], available: newsmap.jp [Accessed: 27 January 2014].

Fig. 2.38 Humble Software Development, *Humble Finance*, 2010, in: humble software development, 2010 [Online], available: www.humblesoftware.com/finance/ [Accessed: 31 August 2014].

Fig. 2.39 P. Cruz, P. Machado, J. Bicker, *Traffic in Lisbon*, 2010, in: P. Cruz, *Visualization et al.*, 2010 [Online], available: pmcruz.com/information-visualization/traffic-in-lisbon-condensed-in-one-day [Accessed: 31 August 2014].

Fig. 2.40 A. Moere, *Particle Animation: Intranet File Usage*, in: A. Moere, *A Model for Self-Organizing Data Visualization Using Decentralized Multiagent Systems*, Advances in Applied Self-organizing Systems, Springer-Verlag London Limited, Part III, p.304, 2008.

Fig. 2.41 A. Moere, *Swarming: Stock Market Quotes*, in: A. Moere, *A Model for Self-Organizing Data Visualization Using Decentralized Multiagent Systems*, Advances in Applied Self-organizing Systems, Springer-Verlag London Limited, Part III, pp.308-309, 2008.

Fig. 2.42 J. Conway, *Game of Life*, 1970, in: *Wikipedia Commons*, 2005 [Online], available: commons.wikimedia.org/wiki/File:Gospers_glider_gun.gif [Accessed: 28 January 2014].

Fig. 2.43 A. Moere, *Cellular Ants: Clustering*, in: A. Moere, *A Model for Self-Organizing Data Visualization Using Decentralized Multiagent Systems*, Advances in Applied Self-organizing Systems, Springer-Verlag London Limited, Part III, pp.315, 2008.

Fig. 2.44 P. Cruz, *An ecosystem of corporate politicians*, 2013, in: P. Cruz, *Visualization et al.*, 2012 [Online], available: pmcruz.com/eco [Accessed: 27 January 2014].

Fig. 2.45 V. Johnston, *FacePrints*, 1994, in: J. Blumberg, New Mexico State University *FacePrints, Maze Solver and Genetic algorithms*, 2005 [Online], available: http://www.cs.nmsu.edu/~ipivkina/Spring05cs579/StudentPres/Faces.pdf [Accessed: 31 August 2014].

Fig. 2.46 E. Altshuler, D. Linden, *Wire Antenna*, 1997, in: A. Marczyk, *The TalkOrigins Archive, Genetic Algorithms and Evolutionary Computation*, 2004 [Online], available: http://www.talkorigins.org/faqs/genalg/genalg.html [Accessed: 31 August 2014].

Fig. 2.47 S. Obayashi, D. Sasaki, Y. Takeguchi, N. Hirose, *Wing Design Solution Comparison*, 2000, in: S. Obayashi, D. Sasaki, Y. Takeguchi, N. Hirose, *Multiobjective Evolutionary Computation for Supersonic Wing-Shape Optimization*, IEEE Transactions on Evolutionary Computation, vol.4, no.2, pp.185, 2000.

Fig. 3.1 B. Fry, *Methodology chart*, in: B. Fry, *Visualizing Data*, 1st ed. Beijing; Cambridge: O'Reilly Media, Inc., p.15, 2008.