Force-directed algorithms for schematic drawings and placement: a Survey

Force-directed algorithms have been developed over the last 50 years and used in many application fields, including information visualisation, biological network visualisation, sensor networks, routing algorithms, scheduling, graph drawing, etc. Our survey provides a comprehensive summary of developments and a full roadmap for state-of-the-art force-directed algorithms in schematic drawings and placement. We classified the model of force-directed algorithms into classical and hybrid. The classical force-directed algorithms are further classified as follows: (a) accumulated force models, (b) energy function minimisation models, and (c) combinatorial optimisation models. The hybrid force-directed algorithms are classified as follows: (a) parallel and hardware accelerated models, (b) multilevel force-directed models, and (c) multidimensional scaling force-directed algorithms. Five categories of application domains in which force-directed algorithms have been adopted for schematic drawings and placement are also summarised: (a) aesthetic drawings for general networks, (b) component placement and scheduling in high-level synthesis of very-large scale integration (VLSI) circuits design, (c) information visualisation, (d) biological network visualisation, and (e) node placement and localisation for sensor networks.

Keywords: Force-directed algorithms; schematic drawing; force-directed placement; information visualisation.

1 INTRODUCTION

Force-directed algorithms have been developed over the last 50 years and adopted in numerous application fields. For example, these include: visualising genetic structures automatically in biology, optimising networks for parallel computer architectures, detecting clusters and hidden patterns in the social sciences, placing and scheduling components for very-large-scale integration circuits (VLSI), and computing undirected/directed networks (graphs) for information visualisation, etc. A schematic drawing is a representation of the elements of a network using simple graphic symbols. Such drawing shows crucial components of the network and the details that are not relevant to the information are omitted [1]. For example, a dot may be used to represent a station in a subway map. In this case, the dot is used to provide key location information to the users without causing any unnecessary visual cluttering. In some application domains, the size of the canvas and the detailed arrangement of the elements in the drawing are constrained by certain technical limits. Force-directed placement [2] is one of the approaches for the node placement in the schematic drawing. The placement of nodes along the edges or in a specific region of the canvas are useful in VLSI applications. According to statistics on annual paper submissions related to force-directed algorithms depicted in Figure 1 (a), force-directed algorithms are very popular and have often been preferred over other algorithms since the 1980s. Figure 1 (a) and Figure 1 (b) show a classification of force-directed algorithms by trends in paper submission and application fields. According to our review, 38% of force-directed algorithm studies relate to schematics and the aesthetics of network visualisation; 30% relate to VLSI applications, with 21% accounted for by placement and 9% by scheduling; in approximately 20% of force-directed algorithm studies, they are applied for social information visualisation; and biological network visualisation and sensor placement and localisation account for 10% and 3%, respectively. These statistics suggest that most applications of force-directed algorithms can be formulated as a problem of network visualisation, which, in turn, can be understood as problem of combinatorial optimisation — to find a visual drawing of an input network topology in a way that optimises functions of interest.

We have adopted a simple approach to classify the papers which are related to force-directed algorithms. The data sources of the papers reviewed in this survey are from ACM Digital Library [3] and Scopus

[4]. The paper submission count of force-directed algorithms classified by application fields is illustrated in Figure 8. First, the papers reviewed in this survey were sorted by publication year. Next, they were categorised into corresponding application domains. The results of the classification are illustrated in Figure 1(c). According to our classification results, studies of force-directed algorithm applications in VLSI have the longest history. The first VLSI study of force-directed algorithms was published in 1965 and this research domain remains popular in 2017. Aesthetics drawing became popular around 1995 and its popularity is ongoing. By contrast, force-directed algorithms for sensor placement and localisation are relatively new research domains. The first publication in this area dates to 2004 and the publication count has increased since 2008. We also found evidence of force-directed algorithmic applications for biological network visualisation dating back to 1995, with publication counts increasing dramatically from 2003 (8 papers per year on average). Finally, studies of force-directed algorithms for social information visualisation have been popular since 2005 and, to date, offer the highest publication counts among all of the research fields.

Each of these applications relates to information visualisation broadly. Information visualisation allows users to make better sense of network relationships than by simply looking at data in tabular form. However, unsupervised visualisation cannot meet these objectives. How network topologies are drawn can significantly affect how viewers understand the network. The layout and position-assignment of visualised network nodes influence how a user perceives network relationships. Identifying visualisations that convey the appropriate information to the user is thus crucial. Filtering and pattern analysis have also been applied for force-directed algorithms to discover insightful relationships and reduce clutter. These methods are especially useful in the visualisation of social data, in which metrics associated with each node are used to understand and identify unexpected network patterns more effectively.

Force-directed algorithms face a number of challenges. Most visualisation problems are NP-hard; as such, approximation methods and heuristics are often proposed, because an almost-global optimum is sufficient for most applications. In addition, force-directed algorithms currently suffer from a number of technical drawbacks. First, they are easy to converge to a localised optima. Second, even the hardware performance has been improved; the running time of force-directed algorithms is still high when producing visualisations for large networks. Third, it is time-consuming to fine-tune the parameters of a large class of networks because the suitable parameters for a particular network class are often disadvantageous for other classes.

Several literature reviews on force-directed algorithms have been published in recent years [5-10]. In [10], Battista et al. presented an annotated bibliography of algorithms for visualisation of graphs. The algorithms reported in their review can be used to visualise various types of graphs such as trees, general graphs, planar graphs, directed graphs, etc. Force-directed algorithms for visualisation of straight-line drawings were also reported in the bibliography. In [5], Gibson et al. reviewed algorithms for force-directed layouts, dimension reduction in graph layout, and multilevel techniques for computational improvements. Gibson et al. also evaluated force-directed algorithms based on aesthetic properties of the drawings such as minimising edge crossings, achieving symmetry, and uniformity on edge nodes, etc. In [7], Tamassia et al. reviewed the algorithms for symmetric graph drawing, tree drawing, spine and radial drawings, circular drawing, rectangular drawing and force-directed drawing, etc. Tamassia et al. also summarised the algorithms and tools used in different application areas such as computer security, education, computer networks, data analytics, graph drawing and cartography, social networks and biological networks. In [11], Battista et al. reviewed the algorithms for force-directed drawing, planer orthogonal –or- straight-line drawings, non-planar drawings, etc. In their review, force-directed algorithms were categorised based on spring and electrical forces, barycenter method, forces simulating graphs theoretic distance, energy functions and magnetic fields. In addition, aesthetic properties such as edge crossings, minimisation of the area of the drawing, minimisation of the length of edges, uniform edge length, uniform bends, symmetric property were also summarised.

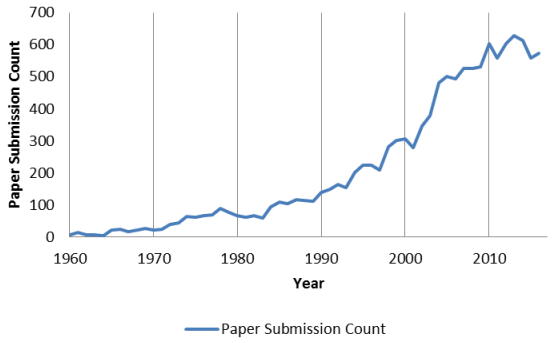
In [6], Kobourov summarised spring systems and electrical forces in graph drawings such as graph theoretic distances approach, stress majorisation, non-Euclidean approaches, and Lombardi spring embedders. They also considered several classical algorithms in spring embedders layouts such as force-directed algorithms, barycentric method, and multiscale methods for dynamic graphs. In [9], Brandenburg et al. compared five force-directed algorithms for drawing graphs in which the positions of the nodes are randomised. Their experiments aimed to evaluate the performance of force-directed algorithms in terms of uniformity in edge length and node distribution. In contrast to previous surveys, in this paper, we provide a comprehensive summary and full roadmap for the state of the art in force-directed algorithms in terms of latest research domains and models including social information visualisation, biological network visualisation, sensor networks, routing algorithms, scheduling, and graph drawing. An overview of the classification of existing force-directed algorithms is also provided in this survey.

In our survey, 230 papers related to force-directed algorithms have been reviewed. To find these papers, we implemented a web mining tool using Java programming language to parse search results from the ACM Digital Library [3] and Scopus [4]. We used four keywords ("force directed algorithms", "force-directed algorithms", "force-directed" and "force directed") to filter relevant papers. The search results from the ACM Digital Library [3] contain the attributes such as authors, title, keywords, abstract and result highlight of papers. The search results from Scopus [4] contain similar attributes except the highlights. Moreover, we applied following filters to remove irrelevant and redundant results in order to improve the accuracy:

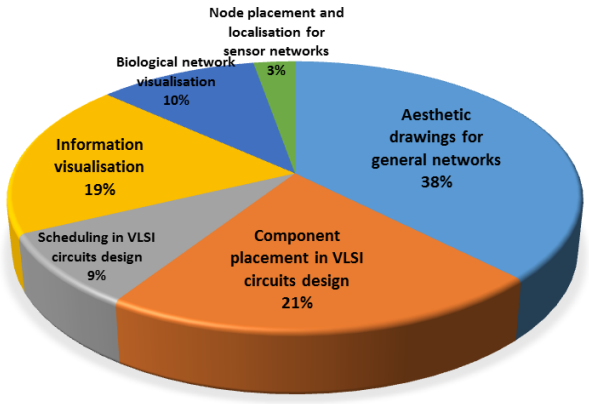
1. The abstract, highlights, the keywords, or the title of the paper must contain at least one of the four keywords used in the searching.
2. Papers returned from partial match were omitted. For example, "They force are applied … directed … algorithm", "… directed …", "force …", "…algorithm".

We also checked the first author and the title of the paper to remove duplicate publications. Figure 2 illustrates state-of-the-art studies and milestones in various force-directed models, including the accumulated force model, the energy function minimisation model, the combinatorial optimisation model, the multilevel model, the multidimensional scaling model and the clustered model. Our findings suggest that many papers are application studies, in which force-directed algorithms are used but without detailed formulation. Application studies usually adopt and/or revise existing force-directed algorithms to achieve the objectives of specified tasks. Because of this, our survey is divided into two parts. For those studies adopting force-directed algorithms to resolve schematic drawings and placement tasks, we first summarise them in our survey in terms of application domains and methods. We then conclude the formulation (model) of notable force-directed algorithms which have been used in application discussed in the first part.

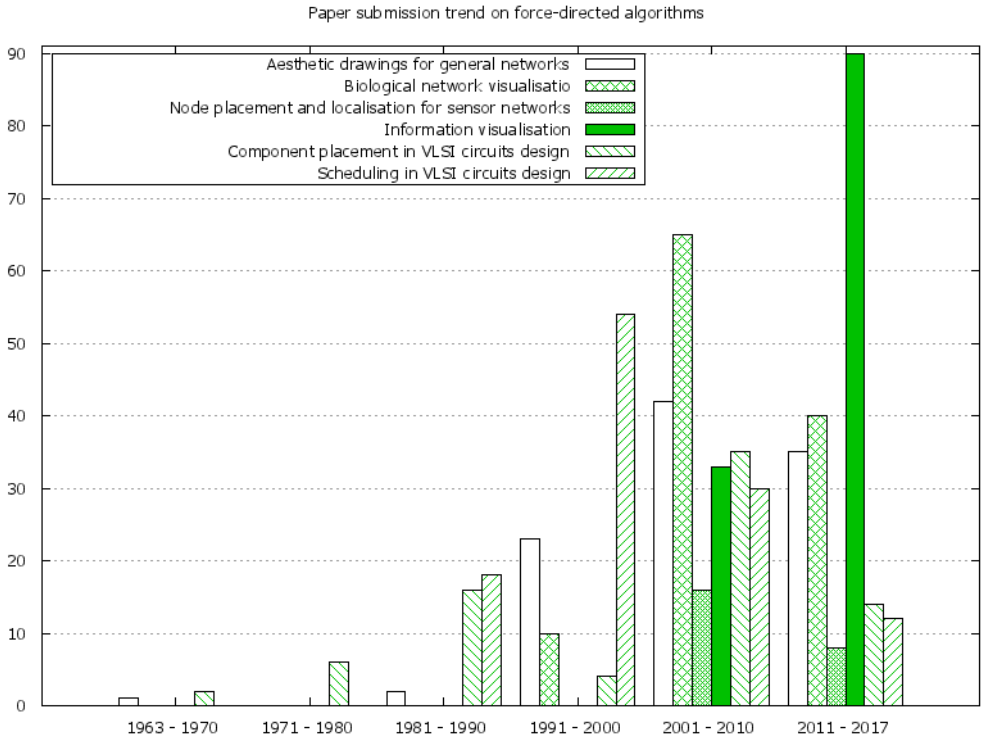
The structure of the survey is as follows: Section 2 presents an overview about the notable force-directed algorithms that have been used most often across the different application domains. Section 3 introduces force-directed algorithms for applications in schematic drawings and placement. Section 4 concludes the survey by summarising patterns across the literature.



(a)



(b)



(c)

Figure 1 (a) Annual paper submission count related to force-directed algorithms; (b) Catalogues of the papers reviewed in this survey; (c) Paper submission trend on force-directed algorithms.

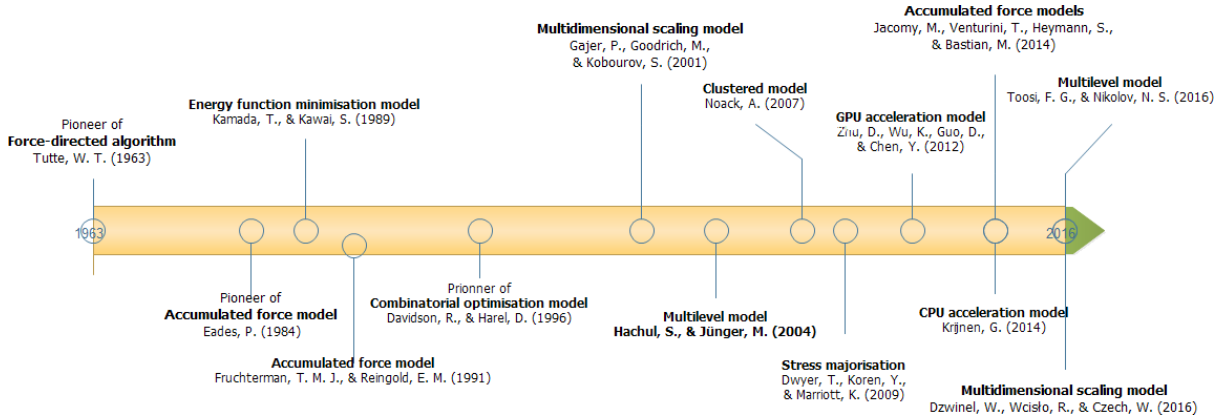


Figure 2 Studies of force-directed models.

1.1 NOTATIONS AND CONVENTIONS

For the purpose of the survey, the notation $G = (V, E)$ represents a network G, including a set of nodes V and edges E between these nodes. The visual drawing of a network is a picture of a network that assigns a position to each node and a curve to each edge. A connected network is a network in which for each pair u, v of nodes, there is always a path between u and v.

2 FORCE-DIRECTED ALGORITHMS

Force-directed algorithms can be divided into classical and hybrid algorithms according to their characteristics and computational modelling. The overview of force-directed algorithms is illustrated in Figure 3. Classical force-directed algorithms are usually based on physical laws, specifically in ways that simulate a spring system. Full descriptions of classical force-directed algorithms are described in section 2.1. Hybrid force-directed algorithms are designed for large and complex networks. These algorithms use heuristics to improve the performance of classical force-directed algorithms. Hardware acceleration and multilevel methods are also popular in improving the performance. Full descriptions of notable hybrid force-directed algorithms are described in section 2.2.

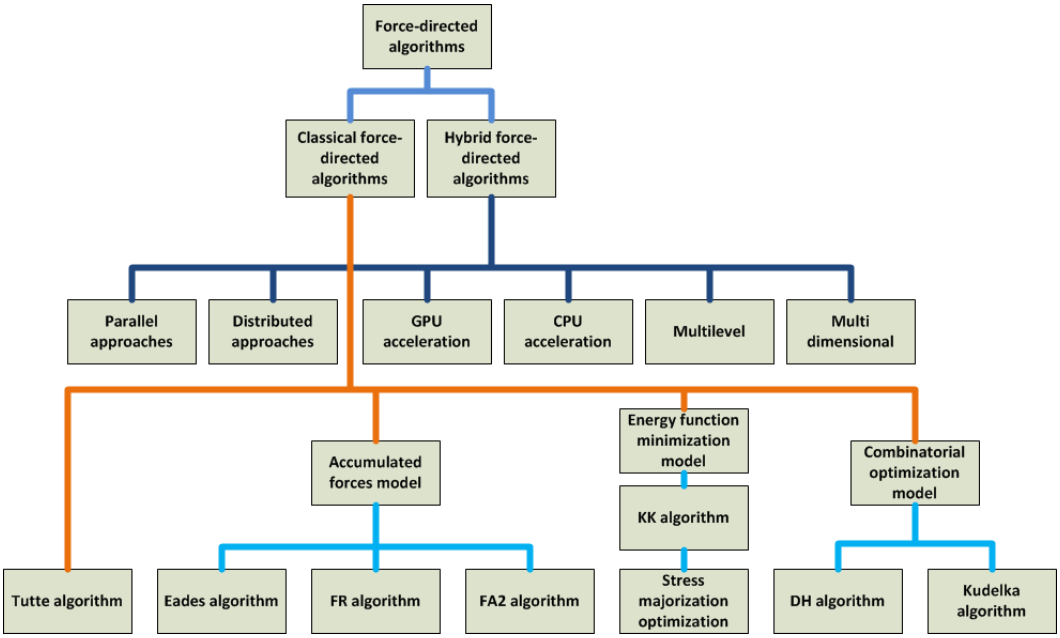


Figure 3 Overview of force-directed algorithms.

The pioneer of force-directed algorithms, the Tutte algorithm, was first proposed in 1963 [197]. The Tutte algorithm is based on the barycentric method [7, 198] and is applicable for tri-connected and planar graphs. A tri-connected graph is a connected graph such that deleting any two nodes results in a graph that is still connected. The force function of a node v of the Tutte algorithm is defined as follows:

$$F(v) = \sum_{u,v \in E} (p_u - p_v) \quad 1)$$

where p_u and p_v are the positions of node u and v. Solving the linear equations from the result of partial derivatives of the force function of Tutte algorithm F can obtain the updated x-coordinate and y-coordinate of nodes. These linear equations are defined as follows:

$$x_v = \frac{1}{deg(v)} \sum_{u,v \in E} x_u \quad 2)$$

$$y_v = \frac{1}{deg(v)} \sum_{u,v \in E} y_u \quad 3)$$

where deg(v) is the number of edges attached to node v. x_v, y_v are the x-coordinate and y-coordinate of node v. An example of the Tutte algorithm is illustrated in Figure 4. Nodes 1, 2, 3, 4 and 5 in Figure 4 form a strictly convex polygon. The Tutte algorithm first selects a strictly convex polygon from the graph, in which all nodes on the convex polygon should have a fixed initial position. Therefore, nodes 1, 2, 3, 4 and 5 are assigned a fixed initial position. The position of remaining nodes (i.e. 6, 7, 8) then can be computed by the Tutte algorithm.

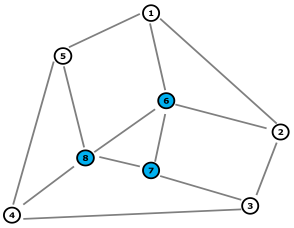


Figure 4 An example of Tutte algorithm.

2.1 CLASSICAL FORCE-DIRECTED ALGORITHMS

2.1.1 Accumulated force models

Accumulated force models follow the simulation of a spring system, in which the length of the spring is proportional to the force exerted by an extended spring. Repulsive and attractive forces are basic forces defined in the accumulated force models. Repulsive force is computed for every node pair and attractive force is computed for every adjacent node [199]. The sum of the values of repulsive and attractive forces for each node are stored in the tempoarry variables, which can be used for updating the nodes'positions. Most accumulated force models follow Hooke's law [200] and the footsteps of Eades' algorithm [26]. Because of this, we first introduce the principle of Eades algorithm in the section 2.1.1.1. We then introduce the successors of Eades algorithm, Fruchterman-Reingold algorithm and ForceAtlas2 algorithm, in sections 2.1.1.2 and 2.1.1.3, respectively.

2.1.1.1 Eades algorithm

The idea of Eades' spring-embedded algorithm is to model a network as a magnetised system with rings representing nodes and the length of edges represented by the spring. Eades [26] was the first algorithm to consider attractive and repulsive forces. The attractive force f_a is applied to nodes that have a direct connection by an edge (i.e. $d(i,j) = 1$), and the repulsive force f_r is applied to nodes that have an indirect connection (i.e. $d(i,j) > 1$). The attractive and repulsive forces of Eades algorithm are defined as follows:

$$f_a(i,j) = C_a \log \frac{d(i,j)}{d_0} \quad (4)$$

$$f_r(i,j) = C_r \frac{1}{d(i,j)^2} \quad (5)$$

where $d(i,j)$ is the distance between node i and j, d_0 is the ideal edge length, and C_a and C_r are the constants. The aim of the algorithm is to find zero-force locations for all nodes to reach a state of equilibrium for the spring system.

2.1.1.2 Fruchterman-Reingold algorithm

The Fruchterman-Reingold (FR) algorithm [2] is based on Eades algorithm [26]. Like the Eades algorithm, the FR algorithm uses two forces, with the attractive force (f_a) and repulsive force (f_r) defined as follows:

$$f_a(d) = \frac{d^2}{k} \quad (6)$$

$$f_r(d) = -\frac{k^2}{d} \tag{7}$$

where d is the distance between two nodes and k is the constant of ideal pairwise distance. For the attraction force, f_a , k can be written as $a \times \sqrt{\frac{W \times H}{n}}$, and can be written as $r \times \sqrt{\frac{W \times H}{n}}$ for the repulsion force, f_r; where W is the width of the canvas, H is the height of the canvas, n is the total number of nodes in the network topology, a is a constant for the attraction multiplier, and r is a constant for the repulsion multiplier.

The FR algorithm is executed iteratively. In each iteration, all of the nodes are moved simultaneously after the forces have been calculated. When updating the position of the nodes, the algorithm adds a 'displacement' attribute to store the position offset of the nodes. At the start of each iteration, the initial values of the displacement for all of the nodes are calculated using the repulsion force (f_r). The algorithm uses the attraction force (f_a) to iteratively update the position of the nodes on every edge. Finally, it updates the position offset of the nodes using the displacement value.

The displacement scale, s, is used as the termination condition of the FR algorithm. When the displacement scale, s, is lower than the threshold value, ε, the algorithm is terminated. When the algorithm is initialised, the value of the displacement scale, s, is set to $\frac{W}{10}$. This value is updated in each iteration according to the iteration count and the maximum number of iterations set by the user.

2.1.1.3 ForceAtlas2 algorithm

ForceAtlas2 (FA2) was proposed by Jacomy et al. [22] to satisfy speed and precision for network visualisation. The algorithm extends the LinLog [32] and FR algorithm [2]. Its authors proposed a revised attractive force based on the LinLog model [32] and defined as follows:

$$F_a(n_1, n_2) = log\big(1 + d(n_1, n_2)\big) \tag{8}$$

where d is the distance between nodes n_1 and n_2. Moreover, a degree-dependent repulsion model was proposed in the FA2 algorithm to reduce the repulsive forces. This repulsion model increases the chances of lower-than-average-degree nodes connecting to higher-than-average-degree nodes.

$$F_r(n_1, n_2) = k \times \frac{(deg(n_1) + 1) \times (deg(n_2) + 1)}{d(n_1, n_2)} \tag{9}$$

where k is a constant of ideal pairwise distance, as used in the FR algorithm [2], d is the distance between nodes n_1 and n_2 and $deg(n)$ is the number of edges associated with the node n, including in- and out-degree edges. In addition, the FA2 algorithm also uses gravitational force and strong gravitational force. Jacomy et al. [22] concluded that strong gravitational force may be useful only for specific types of networks. The definition of these gravitational and strong gravitational forces are defined as follows, respectively:

$$F_g(n) = k \times (deg(n) + 1) \tag{10}$$

$$F_{sg}(n) = k \times (deg(n) + 1) \times d(n) \tag{11}$$

2.1.2 Energy function minimisation model

In contrast to the accumulated force model, the energy function minimisation model uses the spring system to minimise the difference between the visual distance and theoretical graphed distance, and this is accomplished by solving (minimising) an energy function. They do not consider attractive and repulsive forces separately, but rather in conjunction to minimise an energy function. That is, if the visual distance of a pair of nodes is closer than their corresponding theoretical graphed distance, they repel each other; otherwise, they attract each other. The Kamada-Kawai algorithm is the pioneering algorithm for energy function minimisation models. The description of the Kamada-Kawai algorithm is given in section 2.1.2.1 and a technique to improve the energy function minimisation model is summarised in section 2.1.2.2.

2.1.2.1 Kamada-Kawai algorithm

In the Kamada-Kawai (KK) algorithm [31], nodes are placed so that their visual distance within the drawing is proportional to their theoretical graphed distance. As this goal cannot always be achieved for arbitrary network topologies, the key idea behind the algorithm is to use a spring model in such a way that the energy function of the network topology is minimised. The energy function E is:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{i,j} \left(|p_i - p_j| - l_{i,j} \right)^2 \tag{12}$$

where $k_{i,j}$ is the stiffness of a spring between nodes i and j, $l_{i,j}$ is the ideal distance of a spring between nodes i and j, and p_i and p_j are the visual positions of nodes i and j, respectively. That is, the KK algorithm finds a visual position for each pair of nodes i and j, and their Euclidean distance is proportional to $l_{i,j}$. Here, the KK algorithm defines a diameter matrix that stores theoretical graphed distances ($d_{i,j}$) of the nodes. $d_{i,j}$, which represents the hop count between nodes i and j. $d_{i,j}$ is the shortest hop count between nodes i and j. The ideal distance of a spring ($l_{i,j}$) between nodes i and j is defined as follows:

$$l_{i,j} = \frac{L_0}{\underset{i<j}{max} d_{i,j}} \times d_{i,j} \tag{13}$$

where L_0 is the side length of the drawing frame and $\underset{i<j}{max} d_{i,j}$ is the diameter of the network topology. Moreover, the stiffness of a spring between nodes i and j is calculated as follows:

$$k_{i,j} = \frac{K}{d_{i,j}^2} \tag{14}$$

where K is a scaling and $d_{i,j}$ represents the theoretical graphed distances of nodes i and j. The KK algorithm then seeks a visual position for every node v in the network topology and tries to decrease the energy function in the whole network. That is, the KK algorithm calculates the partial derivatives for all of the nodes in the network topology in terms of every x_v and y_v that are zero (i.e., $\frac{\partial E}{\partial x_v} = 0 \; and \; \frac{\partial E}{\partial y_v} = 0, for \; 1 \leq v < n$). However, solving all of these non-linear equations simultaneously is unfeasible because they are dependent on one another. Therefore, an iterative approach can be used to solve the equation based on the Newton-Raphson method. At each iteration, the algorithm chooses a node m that has the largest maximum change (Δ_m). In other words, the node m is moved to the new position, where it can reach a lower level of Δ_m than prior. Meanwhile, the other nodes remain fixed. The maximum change (Δ_m) is calculated as follows:

$$\Delta_m = \sqrt{\left(\frac{\partial E}{\partial x_m} \right)^2 + \left(\frac{\partial E}{\partial y_m} \right)^2} \tag{15}$$

2.1.2.2 Stress majorisation optimisation

In force-directed algorithms such as the KK algorithm [31], visual distance is proportional to the theoretical graphed distance. Stress majorisation optimisation [201-203] is a technique to minimise energy function via majorisation. This technique improves the visual drawing of network topologies iteratively. The principle of majorisation optimisation is to construct a sequence of quadratic forms in which each iteration binds the stress function. The stress function then monotonically decreases (never increases) with every iteration. Thus, a lower value for the energy function is achieved in the same running time [203]. Unlike the KK algorithm, then, the stress function optimised via majorisation is guaranteed to converge [7]. Stress majorisation optimisation is useful for large and clustered networks, especially for applications to social information visualisation [204, 205].

2.1.3 Combinatorial optimisation model

Combinatorial optimisation models are probabilistic algorithms, often inspired by evolutionary mechanisms. Simulated annealing, differential evolution and genetic algorithms use a number of measures to improve a candidate solution and to optimise a problem iteratively. Although these algorithms share many similar properties, they still have distinctive features, including population determination, strategies to search the solution state space, etc. [206].

2.1.3.1 Davidson-Harel algorithm

The process of simulated annealing is inspired by the physical cooling process of molten materials. Molten steel will crack and form bubbles that make it brittle if cooled too quickly. The steel must therefore be cooled evenly for a better result — a process known as annealing in metallurgy [7, 207, 208]. The Davidson-Harel (DH) algorithm [29] uses a simulation of the annealing process to prevent nodes from moving too close to non-adjacent edges and to minimise edge crossings. An energy value E, attraction force f_a and repulsion force f_r are used in the simulation. The energy value (E) is the sum of all attraction forces (f_a) and repulsion forces (f_r) which can be calculated as follows:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_a\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right) + f_r\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right) \quad (16)$$

A node i is randomly selected from the network on initialisation. The DH algorithm then creates a temporary node j, and assigns a position to the node based on the position of node i. Therefore, a new energy value E' can be calculated using the position of node j and other nodes within the network.

$$E' = \sum_{v,i \in V, j \notin V, v \neq i} f_a\left(\sqrt{(x_v - x_j)^2 + (y_v - y_j)^2}\right) + f_r\left(\sqrt{(x_v - x_j)^2 + (y_v - y_j)^2}\right) \quad (17)$$

Moreover, the DH algorithm obeys the rules of the Boltzmann distribution when the liquid is cooled slowly [209]. If $E' - E \leq 0$, then E' is used as the energy of the next iteration, as E' has lower energy value. If $E' - E > 0$, a probability equation is used to determine whether to use the new energy E' in the next iteration. The probability equation is defined as follows:

$$p = e^{-\frac{(E'-E)}{k \times T}} \quad (18)$$

where T is the temperature variable and k is the Boltzmann constant. If the probability p is less than the threshold φ, then the new energy E' is accepted; otherwise, the old energy E will be used in the next iteration.

2.1.3.2 Kudelka algorithm

The Kudelka algorithm [210] is a force-directed algorithm that aims to find a low-dimensional representation of the high-dimensional network. This allows the high-dimensional network to be visualised in low-dimensional (e.g. two- or three-) space. Sammon's mapping [211] and the differential evolution method are used in the Kudelka algorithm. Differential evolution is a population-based optimiser. It evolves a population of real encoded vectors in which the initial values of vectors are randomly chosen from within a predefined range. Differential evolution generates new vectors and operations using the real encoding of candidates. As a result, new vectors are perturbed and scaled from the existing vectors of the population. The objective of the Kudelka algorithm is to minimise the projection error function E, which is defined as follows:

$$E = \frac{1}{\sum_{i<j}^{m} d_{ij}^*} \sum_{i<j}^{m} \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*} \quad 19)$$

where d_{ij}^{*} is the distance between X_i and Y_j. The distance between corresponding vector Y_i and Y_j in lower dimensional space is denoted as d_{ij}.

2.2 HYBRID FORCE-DIRECTED ALGORITHMS

Several studies have used heuristic techniques to improve the performance of force-directed algorithms and reduce execution time, enabling the algorithms to visualise large and complex networks in an efficient manner. For example, the multilevel technique simplifies networks through network abstraction processes. Distributed force-directed algorithms use parallel computing and hardware acceleration to reduce execution time for parsing large networks. The multidimensional scaling technique is useful for visualising networks' meaningful underlying dimensions. State-of-the-art studies of these heuristics are discussed and summarised in the following sections.

2.2.1 Parallel and hardware accelerated force-directed algorithms

The major principle of parallel computing is to solve a computational problem using multiple resources simultaneously [212, 213]. Generally, parallel computing involves the following steps:

1. A computational problem is first broken into smaller pieces of executable content that can be solved concurrently.
2. Each piece of executable content will be further broken down into a series of instructions for the Central Processing Unit (CPU) or Graphics Processing Unit (GPU).
3. Instructions from every piece of executable content are executed simultaneously on different CPU or GPU.
4. An overall coordination mechanism is used. When a task has completed the execution of instructions, it sends an acknowledgment to the coordinator before sending the result to the receiving task.

Most parallel computing frameworks [214-216] for force-directed algorithms are based on the accumulated force model. For example, the GPU parallel computing framework [217] was proposed for identifying the k-nearest neighbours, the results of which were then utilised to speed up the FR algorithm [2]. A distributed force-directed algorithm in an open source distributed computing framework called Giraph1 [218] was implemented in Amazon's cloud computing infrastructure PaaS (Platform as a Service) [219]. Arleo et al. [218] claimed that the algorithm can process networks with up to million edges. A parallel FR algorithm [2] based on Open Computing Language (OpenCL) was proposed by Krijnen [220] and Wang et al. [221]. OpenCL programs can be executed across heterogeneous platforms with modern CPUs, GPUs, and microprocessor designs [222]. There are also parallel force-directed algorithms [223-225] based on the Message Passing Interface (MPI). MPI is defined by a group of parallel computing vendors and applications specialists[2] as a specification for a standard library for message passing in distributed computing.

2.2.2 Multilevel force-directed algorithms

The multilevel technique for force-directed algorithms involves concepts from network abstraction and can be divided into two main phases. In the first phase, called 'coarsening', the original network is split into a sequence of coarse networks with decreasing sizes. This simplifies the combinatorial structure of the network by selecting the coalescent pairs of adjacent nodes to construct a new network. The selection process is repeated recursively to abstract a sequence of such coarse networks. The process of energy optimisation (minimisation) is then performed across these coarse networks such that they are optimised using the global properties from the original network. The second phase is called refinement and involves successive drawings of fine networks

[1] http://giraph.apache.org/
[2] http://mpi-forum.org/

computed from the smallest coarse networks. Finer networks are optimised using the locally determined properties from the related coarse network. As a result, it can decrease running time because the energy minimisation process considers only a small amount of neighbourhoods at once [226]. Many studies have proposed using the multilevel technique for force-directed algorithms [227-230]. There are also studies that extend the multilevel technique to the classical force-directed algorithms such as multilevel KK algorithm [30] and multilevel FR algorithm [24].

2.2.3 Force-directed algorithms with multidimensional scaling

High-dimensional data usually have a large number of variables instead of a large number of duplicated records. The multidimensional scaling technique is widely used in force-directed algorithms for high-dimensional data reduction. The objective of the multidimensional scaling technique is to find meaningful underlying dimensions so that observed similarities and dissimilarities from the investigated networks can be discerned easily. The principle behind multidimensional scaling was developed by Torgerson [231] which uses the distance of edges as a metric. Nodes are projected into a smaller space that satisfies the constraint of the metric (the distance of edges). Many studies have adopted multidimensional scaling for force-directed algorithms to visualise high-dimensional data in which the distances between pairs of data are preserved [201, 232-239]. Multidimensional scaling is also useful for energy function minimisation modelling, as it can improve the layout of networks with high-degree nodes. Dwyer et al. [240] and Dzwinel et al. [241] proposed a multidimensional scaling KK algorithm [31] with the use of stress majorisation optimisation. The energy function proposed by Dzwinel et al. [241] is defined as follows:

$$E = k_{nn} \sum_{i}^{N} \left(\sum_{j \in O_{nn}(i)}^{nn} {d_{ij}^{n}}^2 + c \times \sum_{k \in O_{rn}(i)}^{rn} (1 - d_{ik}^{n})^2 \right) \quad (20)$$

where k_{nn} and c are constants and configured by users. d_{ij}^{n} is the distance of node i and j in the visual drawing. $O_{nn}(i)$ is the nearest neighbourhood of node i (i.e. hop count equals to 1). $O_{rn}(i)$ is the random neighbourhood of node i (i.e. hop count greater than 1).

3 APPLICATIONS OF FORCE-DIRECTED ALGORITHMS

This section reviews five categories of application domains in which force-directed algorithms have been adopted: (a) aesthetic drawings for general networks, (b) component placement and scheduling in high-level synthesis of very-large scale integration (VLSI) circuits design, (c) information visualisation, (d) biological network visualisation, and (e) node placement and localisation in sensor networks.

3.1 FORCE-DIRECTED ALGORITHMS IN AESTHETIC DRAWINGS FOR GENERAL NETWORKS

Force-directed algorithms can be used to produce schematic drawings from network topology alone, even without additional information about its nodes and edges. However, many applications of force-directed algorithms involve an implicit aesthetic problem in how to schematise topological renderings. The importance of such schematics is that its depiction can significantly influence how the topology is understood. For example, what are the aesthetic properties of the most coherent schematics? How can the aesthetic quality of schematics be measured? To understand these questions, we need to clarify the characteristics and objectives of a schematic. The fundamental factor is its layout. For example, in the polyline drawing (see Figure 5 (a)), each edge is a polygonal chain. Whereas in the straight-line drawing (see Figure 5 (b)), each edge is a straight-line segment. In the orthogonal drawing (see Figure 5 (c)) [8], each edge represents a horizontal and vertical segment. Numerous visualisation tools have been implemented for visualising networks in different layouts, most developed for

straight-line drawing, such as GraphED3 [12], COMAIDE [13], LayoutShow [14], Graphael [15] and OpenOrd [16]. A visualisation tool based on orthogonal drawing is also proposed in [17].

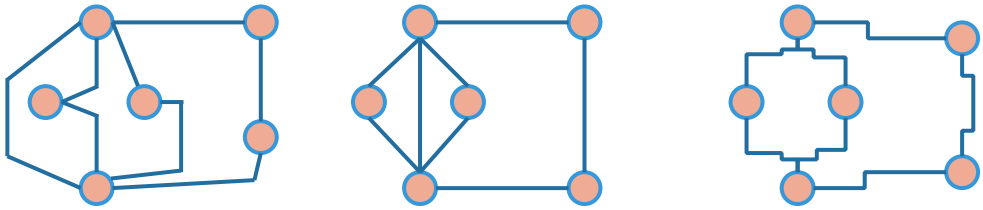


Figure 5 (a) Polyline drawing (b) Straight-line drawing (c) Orthogonal drawing

Creating aesthetically appealing schematics has the practical aim of revealing a structure's pattern, rather than being merely a quest for the beautiful [18]. Therefore, researchers have defined the properties of a schematic based on its fundamental factors. Force-directed algorithms can be used to produce schematics that adhere to the properties of aesthetic drawing [9, 19, 20]. The properties of aesthetic drawing include : 1) edge lengths should be uniform; 2) the number of edge crossings should be minimised; 3) the size of crossing angles should be uniform; 4) the crossing angle should be minimised; 5) the standard deviation of edge length should be low; 6) the angle formed by any two neighbouring edges should be minimised; 7) the number of bends in polyline edges should be minimised; 8) nodes and edges should be affixed to an orthogonal drawing; and 9) the network should be represented as symmetrically as possible. In [21], Tunkelang proposed a force-directed approach for drawing undirected graphs. It is based on the accumulated force model that includes repulsive and attractive forces. Repulsive forces are computed between any two nodes and attractive forces are calculated between two adjacent nodes. Repulsion among nodes are used to avoid situations where nodes are placed too close to each other. Attraction forces are used to prevent nodes from being too far away from each other. According to the principles of the accumulated force model [22], nodes pull far away from each other if they are not adjacent. Besides, the model tries to maintain uniform edge lengths among adjacent nodes to minimise edge crossings. The repulsive and attractive forces of the proposed algorithm are defined as follows:

$$f_r(d) = \frac{w_r}{d^2} \tag{21}$$

$$f_a(d) = w_a d \tag{22}$$

where d is the length of edge and w_r and w_a are constants. The objective of the algorithm is to find an optimal value d so that the sum of attractive and repulsive forces (i.e. $f_r(d) + f_a(d)$) is minimal. In addition, a force-directed algorithm was also proposed to produce schematics based on the fitness function of a genetic algorithm (GA) [23]. A number of studies have adopted similar approaches in the literature. Due to the page limit, we summarise them in terms of the models used and the property of the aesthetic drawing in Table 1.

Table 1 Forced-directed algorithms for aesthetic visual drawings.

Catalogue	Property of aesthetic drawing	Adopted by proposed force-directed algorithms	Models used in force-directed algorithms
Node	Distribute nodes evenly	[2], [22], [24],	Accumulated force model

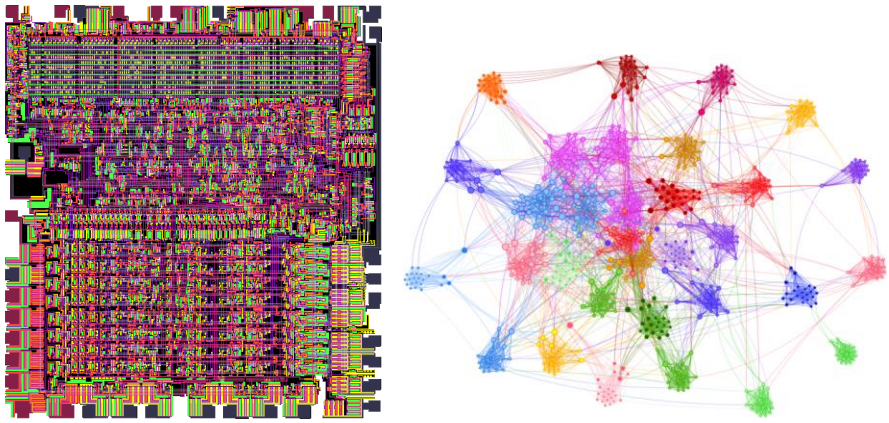
[3] http://www3.cs.stonybrook.edu/~algorith/implement/graphed/implement.shtml

		[25], [26], [27], [28]	
		[29]	Combinatorial optimisation model
		[30]	Energy function minimisation model with a multiscale approach
		[31]	Energy function minimisation model
		[23]	Energy function minimisation model with a fitness function in GA
	Cluster similar nodes	[32]	Energy function minimisation model
	Nodes should not overlap	[33]	Accumulated force model
	Nodes that are not adjacent should be far away from each other	[21]	Accumulated force model
		[29]	Combinatorial optimisation model
Edge	Minimise edge crossings	[2], [21], [22], [24], [26], [27], [28], [34]	Accumulated force model
		[29]	Combinatorial optimisation model
		[35]	Multilevel force-directed algorithm
		[36], [37]	Energy function minimisation model
	Minimise edge bends	[35]	Multilevel force-directed algorithm
	Keep edge lengths uniform	[2], [21], [22], [24]	Accumulated force model
		[31]	Energy function minimisation model
		[23]	Energy function minimisation model with an additional fitness function of genetic algorithm
		[30],	Energy function minimisation model with a multiscale approach
	Minimise edge length	[28]	Accumulated force model
		[23]	Energy function minimisation model with an additional fitness function of genetic algorithm
all layo	Display of symmetries	[25], [26]	Accumulated force model

		[31], [37]	Energy function minimisation model
		[30]	Energy function minimisation model with a multiscale approach
		[23]	Energy function minimisation model with an additional fitness function of genetic algorithm
	Maximise the angles among incident edges	[28], [38]	Accumulated force model
		[35]	Multilevel force-directed algorithm
		[36]	Energy function minimisation model
		[23]	Energy function minimisation model with an additional fitness function of genetic algorithm
		[39]	
	The angles between edges incident on the same node should be as uniform as possible	[28], [39]	Accumulated force model
		[36]	Energy function minimisation model
		[23]	Energy function minimisation model with an additional fitness function of genetic algorithm
	Orthogonality	[17], [40]	Accumulated force model with an additional octilinear magnetic force [41] for orthogonal drawing
	Minimise the size of visual drawing	[37]	Energy function minimisation model

3.2 FORCE-DIRECTED ALGORITHM IN COMPONENT PLACEMENT AND SCHEDULING IN VLSI CIRCUITS DESIGN

Technical Terms such as ‘module’, ‘cell’, ‘pin’ and ‘component’ are widely used in the studies of very-large-scale integration (VLSI) circuits. They are similar to the concept of nodes in graph theory. To make the terms consistent in this survey, we use the term ‘node’. Force-directed placement algorithms and force-directed scheduling are widely used in the design and manufacturing for VLSI circuits. An example of components from a VLSI circuit board is illustrated in Figure 6 (a) (generated by the visual5602 simulator [42]). The roadmap and approaches for these techniques are discussed in the following subsections.



(a) (b)

Figure 6 Visualisations of (a) components of a VLSI circuit (b) a clustered network.

3.2.1 Force-directed placement algorithms

The nodes in VLSI circuits can be integrated circuits, transistors, resistors and capacitors. The interconnection topology of the VLSI circuits is known. The objective of force-directed placement algorithms in this context is to determine the optimal location of every node with respect to every other node such that the length of edges in the interconnection topology is minimised [43]. Force-directed placement algorithms can obtain fairly non-overlapping placements on circuit boards without the use of additional means of optimisation [44] and, as such, have proven popular in applications to VLSI circuit boards since the 1960s [45-51].

3.2.1.1 Pioneer approaches

Fisk and Isett [45] pioneered a system called ACCEL using two forces (i.e. attractive and repulsive forces) for the placement of nodes. Urban et al. [47] proposed a system called SHARPCLAW using similar forces [45]. Quinn and Breuer [46] and Quinn Jr [43] proposed similar systems based on Hooke's Law, with repulsive and attractive defined as follows:

$$F_r(u,v) = -\frac{K_r}{\sqrt{\frac{(x_u - x_v)^2}{(w_u - w_v)^2} + \frac{(y_u - y_v)^2}{(h_u - h_v)^2}}} \tag{23}$$

$$F_a(u,v) = -K_a\sqrt{\frac{(x_u - x_v)^2}{(w_u - w_v)^2} + \frac{(y_u - y_v)^2}{(h_u - h_v)^2}} \tag{24}$$

where K_r and K_a are the constants for repulsive and attraction forces, x_u, y_u are the x-coordinate and y-coordinate of the node u. w_u and h_u are the width and height of the node u.

3.2.1.2 Modern approaches

Numerous notable force-directed placement algorithms and open-source systems have been developed since the 1990s. Most are based on solving a quadratic cost function to optimise node placement and achieve minimal edge lengths on the circuit board [52]. Force-directed relaxation methods are often used to solve the quadratic cost function. Force-directed relaxation is an iterative method in which nodes are either assigned random or fixed locations on initiation. One node is then selected at each iteration and moved to a target point determined by the forces or cost functions defined in the force-directed placement algorithms [53]. Popular

algorithms include Kraftwerk [54], Kraftwerk2 [55], FAR [56], mFAR [57], FDP [58, 59], FastPlace [60], FastPlace 3.0 [61], RQL [62], SimPL [63], etc., the objective of which is to evenly distribute electromechanical components (nodes) on the circuit board, minimise the wire (edge) length and produce an overlap-free layout [64]. For example, the Kraftwerk [54] algorithm formulates the quadratic cost function F defined as follows:

$$F = \sum_{u,v \in V} \frac{1}{2}\left(w_{uv,x} \times (x_u - x_v)^2 + w_{uv,y} \times (y_u - y_v)^2\right) \tag{25}$$

where x_u and y_u are x-coordinate and y-coordinate of node u. $w_{uv,x}$ is the weight of edge uv on x-axis (horizontal), $w_{uv,x}$ is the weight of edge uv on y-axis (vertical). The weight used in the x-axis and y-axis from equation (25) is different because the node (electromechanical component) placed on the VLSI circuit board is quadrilateral. We also found several extensions of the Kraftwerk algorithm [54] proposed for application in VLSI circuits [65-68]. A similar algorithm called FastPlace was proposed by Viswanathan and Chu [60]. FastPlace is also based on a force-directed relaxation precept that aims to evenly distribute nodes on the circuit board. This can be done by minimising the cost function, which is similar to equation (25). In contrast to others, Viswanathan and Chu [60] applied a post-processing technique called 'cell shifting' to reallocate the positions of nodes that overlap as a result of force-directed placement. Pan et al. [61] also proposed an improved extension of the FastPlace algorithm, called FastPlace 3.0, which adopts a multilevel technique and uses congestion constraints [69] to place nodes evenly.

3.2.1.3 **Partitioning and clustering based approaches**

Goto [70] used a force-directed placement algorithm to divide nodes on the circuit board into two parts: an initial placement and an iterative improvement [71]. Nodes have pre-assigned (fixed) positions in the initial placement, and the force-directed algorithm calculates node locations during the improvement phase only. An algorithm based on [70] was proposed by Chang [72]. The objective of the algorithm is to find optimal regions on the circuit board to place nodes. The algorithm extends the median formulation proposed by [70] which identifies optimal regions and then applies a force-directed algorithm to calculate nodal positions within each optimal region. A force-directed placement algorithm based on clustering was also proposed by Odawara et al. [73] in which 'seed elements', such as CPU and ROM from the circuit board, are first identified. Nodes close to seed elements are then grouped together to construct clusters. Finally, the relative position of each cluster is calculated by the force-directed algorithm. A similar system adopted a clustering technique was suggested by Alupoaei and Katkoori [74]. In Alupoaei's algorithm, clique partitioning heuristics [75] were used to cluster nodes and a force-directed algorithm based on Hooke's Law [46] was used to determine node placement and to minimise edge lengths on the circuit board. In [76], Vorwerk and Kennings [76] introduced a multilevel clustering algorithm to extend the algorithm proposed by [59]. The Hybird First Choice [77] clustering method was used in the Vorwerk and Kennings's algorithm in order to improve node placement.

3.2.1.4 **Fixed-points and pseudo edges additional approaches**

The placement of standard cells is another major application in VLSI circuits. Standard cells function as nodes with standard heights but varying widths. Numerous studies focus on the placement of standard cells. For example, some have used the cost function from the Kraftwerk algorithm [54] to determine the placement of standard cells [65]. Chou and Lin [78] located standard cells by adding additional pseudo-edges on the circuit board. In this algorithm, critical paths on the circuit board are first identified. Pseudo-edges will then attach to nodes that are close to critical paths to pull the position of nodes closer to the critical paths. All pseudo-edges are removed when the placement is completed. In addition, Hu and Marek-Sadowska [56] introduced an algorithm called FAR to add additional fixed-points (nodes). A fixed-point is a pseudo-node connected to a real node on a circuit board. Three types of fixed points are defined by [56]: controlling fixed points are used to keep the placement of a node unchanged, perturbing fixed points are used to disturb the current placement, and constraining fixed points are used to restrict the movement of a node. In another example, a flat force-directed placement algorithm called SimPL was proposed by Kim et al. [63] that does not rely on clustering. SimPL has a

range of variants [79-82], all of which adopt a top-down geometric partitioning method called a look-ahead legaliser [83] to remove nodal overlap. SimPL's variants add fixed-points and pseudo-edges to produce even nodal distributions, for which the concept of fixed-points and pseudo-edges are adopted from the FAR algorithm [56]. In addition, a multilevel force-directed placement algorithm based on the energy function minimisation model [65] and fixed-point addition [56] was proposed by Hu and Marek-Sadowska [57].

3.2.1.5 **Heuristic and application domain dependent approaches**

Forbes [84] proposed a heuristic approach to accelerate the force-directed placement algorithm proposed by Fisk and Isett [45]. The objective of the heuristics is to reduce the total number of iterations of the force-directed placement algorithms. The movements of nodes during previous iterations are used to predict the position of a node in one or more future iterations. Spindler et al. [55] proposed an extension called Kraftwerk2, which is based on previous work [85]. The objective of the Kraftwerk2 algorithm is to balance the density of nodes and reduce and/or prevent any unused area (free space) of circuit board (i.e. save the space of circuit board). Two types of nodes are defined in the Kraftwerk2 algorithm. One has a fixed initial position (i.e. FN) and the other does not (i.e. MN). Only positions of the MN need to be determined in the Kraftwerk2 algorithm. Moreover, three forces are defined in the algorithm: Net Force F_V^{net}, Move Force $F_{V,u}^{move}$ and Hold Force F_V^{hold}. The force equation of the Kraftwerk2 algorithm is the sum of the three forces and defined as follows:

$$F = F_V^{net} + F_{V,u}^{move} + F_V^{hold} \tag{26}$$

The forces of the Kraftwerk2 algorithm use concepts from a generic supply and demand system [86]. Spindler et al. [55] stated that the Net Force F_V^{net} is used to minimise edge length. However, nodes will overlap when the edge length is too short. Therefore, Move Force $F_{V,u}^{move}$ and Hold Force F_V^{hold} are added to the Kraftwerk2 algorithm to compensate the Net Force F_V^{net} as a way to reduce nodal overlap. Interested readers can refer to Nam and Cong [86] for detailed definitions and explanations about the generic supply and demand system.

Heuristic approaches were also used for the placement of standard cells. A heuristic force-directed algorithm for the placement of standard cells was proposed by Hur et al. [87]. Congestion removal heuristics [69] are applied in Hur et al.'s algorithm to remove nodal overlaps. Additionally, a force-directed placement algorithm for determining the location of standard cells in 3D ICs (integrated circuits) away from high-temperature areas was also introduced [88].

Floor-planning is an application in VLSI closely related to placement. The goal of floor-planning algorithms [89, 90] is to develop a placement plan to decide topological proximity and the appropriate shapes and orientations of each block. A placement algorithm using the maze searching technique [91] was proposed by Mo et al. [92]. The algorithm was designed to minimise edge lengths on a circuit board. The maze searching technique is able to find the shortest path from a given node to another given node. The approach proposed by Mo et al. applies force-directed algorithms to the placement of nodes first and then uses the maze searching technique to re-route paths (edges) on the circuit board and minimise edge lengths.

Minimising the timing delay of circuits is another important task for VLSI. Force-directed placement algorithms based on Kraftwerk [54] proposed by Rajagopal et al. [93] aim to optimise the edge lengths and minimise the timing delay on the circuit board. A similar approach was proposed by Saxena and Halpin [94] to optimise the timing delay of circuits, which improves the repeater insertion technique [95] by using a force-directed approach based on Kraftwerk [54]. Repeater insertion techniques can reduce the time delay associated with long wire lines in circuit. In addition, Goplen et al. [96] proposed an algorithm to reduce repetitions during placement in which weightings [59] are used to reduce the repeater count. In [96], the cost function is adopted from Goplen's algorithm [65].

Besides timing delay minimisations, density information can also be used to improve force-directed placement algorithms [62, 97]. For example, improved versions of cell-shifting techniques were proposed by Viswanathan et al. [62]. These techniques adopted a Density-Aware Module Spreading algorithm [98] and

extended the cost function of quadratic optimisation from [65] to improve the placement of nodes on circuit boards. Viswanathan et al. [62] used the density information to prevent nodes from being placed on areas that already contain high densities of edges and nodes.

Mixed-size integrated circuit (IC) design, in which the network contains a large number of nodes and macros, is also widely used in VLSI. In most cases, the magnitude (size) of macro force is larger than the size of nodes [99]. For this reason, placement algorithms should use smoothing approaches to place both nodes and macros on the chip areas simultaneously. A force-directed placement algorithm called FDP was proposed for the placement of mixed-size integrated circuits [58, 59]. The algorithm uses a dynamic weighting [100] of spreading forces. The cost function of FDP is defined as follows:

$$F = \sum_{u,v \in V} \frac{a_{uv}}{|p_u^{i-1} - p_v^{i-1}|} (p_u^i - p_v^i)^2 \tag{27}$$

where a_{uv} represents the weight of the edges connecting node u and v. p_u^i and p_u^{i-1} are the position of node u at iteration i and $i-1$, respectively. The objective of FDP algorithms is to minimise the cost function in equation (27).

Placement algorithms for 3D Field Programming Gate Array (FPGA) [101] consisting of multiple two-dimensional layers have become popular in recent studies. A low temperature simulated annealing method [102] can be used to determinate the final 3D layer from the two-dimensional layers. The latest 3D FPGA applications can be found in force-directed algorithms, such as those using the force-directed placement algorithm to minimise the edge lengths on each two-dimensional layer [103]. Integrating optical devices into the electronic communication system NoC (Networks-on-Chip) [104] is one example. The PLATON algorithm is proposed by [105] to place overlap-free Photonic Switching Elements (PSEs) on the circuit board. PSEs are components used in optical networking.

3.2.2 Force-directed scheduling algorithms

Force-directed scheduling algorithms are useful in High Level VLSI Synthesis systems [106-109]. An algorithm's description of a design behaviour can be interpreted by high-level synthesis [108]. For example, the context of encoding algorithms can be interpreted by high-level synthesis such that the hardware encoder/decoder algorithm can be implemented on integrated chips. Force-directed scheduling algorithms schedule instructions and operations for high-level synthesis to optimise the distribution of operations and reduce resource expenditure.

The initial force-directed scheduling algorithm was first proposed by Paulin and Knight [110] and, like other force-directed algorithms, it obeys Hooke's Law in physics. Paulin et al.'s algorithm attempts to balance the distribution of operations by decreasing concurrency of operations that make use of the same hardware resources. In the initial version of force-directed scheduling for the behavioural synthesis, proposed in Paulin and Knight [111], operations are divided into a number of steps, all of which aim at reducing the number of data buses, storage units and functional units while maintaining the concurrent operations assigned to them without lengthening the total execution time. Paulin and Knight [112] presented a force-directed scheduling algorithm to minimise interconnected costs of register allocation in high-level synthesis. Variants and extensions based on this pioneering work have been developed and reported in [113-122]. Classical scheduling has been used to minimise resources by finding a feasible schedule τ that minimises the resource costs. The schedule of classical scheduling is defined as follows:

$$f(\tau) = \sum_{r \in R} w_r \, {}^{max}_{t \in T} N_r(\tau, t) \tag{28}$$

where R is a set of resource types in which $r \in R$. w_r is the cost of a resource type r and t is the span of time required of a schedule τ. However, solving equation (28) is a NP-complete problem. Therefore, Verhaegh et al.

[123] presented an iterative approach for the forced-directed scheduling algorithm used in PHIDEO [124] silicon compilers. The cost function of their iterative approach is defined as follows:

$$f(r) = \sum_{r \in R} w_r u_r + \sum_{r \in R} w_r \, {}^{max}_{t \in T}(N_r(\tau, t) - u_r) \tag{29}$$

$$u_r = \frac{1}{m} |\{i \in O | r_i \in r\}| \tag{30}$$

where u_r is a constant based on the average number of operations for resource type r over a schedule in which m is the time span on a given schedule. w_r is the cost for a resource type r. $N_r(\tau, t)$ is the number of operations of resource type r scheduled at time t in schedule τ. The objective is to minimise the cost function $f(r)$. Verhaegh et al. [125] also presented an iterative force-directed scheduling algorithm which reduces the time span of an entire operation schedule, as used in the silicon compiler PHIDEO [124].

Behavioural synthesis systems are generally designed for single tasks. Lee et al. [114] proposed a heuristic force-directed scheduling algorithm for multi-thread, real-time and multi-tasking synthesis systems. Lee et al.'s algorithm is based on the A^* search technique and the force-directed scheduling algorithm proposed by Paulin et al. [111]. Multi-tasking synthesis systems contain a set of k processors and a set of n periodic real-time operations. The principle is to assign each operation to one of the processors in such a way that all operations can be scheduled within their time constraints. Lee et al.'s algorithm used the A^* search technique to select processors that minimise the cost and satisfy timing constraints. Moreover, Abdel-Kader [115] used a force-directed scheduling algorithm derived from [111] to optimise loop scheduling in high-level synthesis. Loop scheduling is designed for repetitively performing a set of operations that functions similar to a loop in programming. Some extensions of [111] work were also proposed for reconfigurable architectures. For example, a force-directed scheduling for schedule operations in NATURE [126] was proposed by Zhang et al. [116]. NATURE is a hybrid nano/CMOS reconfigurable architecture. Force-directed scheduling algorithms are also useful for Dynamic Reconfigurable FPGAs (DRFPGAs) [117], owing to overlaps in the logic of DRFPGAs as time-multiplexed. Because of this, DRFPGAs need to be partitioned into multiple sub-circuit boards, thus possibly resulting in different execution times because sub-circuit boards are executed in parallel. Force-directed scheduling algorithms can be used to partition sequential circuits to optimise feasible partitions that reduce the logic and communication component costs while maintaining maximal throughput.

Force-directed scheduling algorithms for power optimisation problems in VLSI high-level synthesis systems have been popular since 2000. These algorithms are based, again, on the work of [111]. For example, some have used a force-directed scheduling algorithm to optimise power consumption while adhering to the resource and latency constraints in a behavioural synthesis system [119]. Gupta and Katkoori [120] also used a force-directed scheduling algorithm to optimise power consumption at the behavioural synthesis system. They reduced the overall dynamic power by reducing switched capacitance component usage during VLSI circuit design. Moreover, Allam and Ramanujam [121] proposed a force-directed scheduling algorithm for power optimisation that minimises the peak and average consumption. This can be done by assigning the smallest possible input voltage to every operation in a way that minimises power consumption.

Advanced driver assistance functions for intelligent automotive systems, such as predictive break assistants, adaptive cruise control and adaptive lane assistance are designed for processing sensor data. Schönwald et al. [122] proposed a force-directed scheduling algorithm for advanced drivers to map processes to processor cores with time and resource constraints. The objective of this work is to reduce the communication latency and increase the throughput to process sensor data. Similarly, Schönwald et al. [127] proposed a force-directed scheduling algorithm to consider shared memory architectures during the mapping of software processes on multiprocessor system-on-chip (MPSoC) cores. Schönwald et al. [127] suggested the use of smFDM (shared memory aware force-directed mapping) to determine the placement of processor cores.

Therefore, communication conflicts and memory access conflicts are reduced or even avoided. Force-directed scheduling algorithms were also proposed by Omnés et al. [118] to schedule real-time tasks to be executed on embedded multimedia systems. The work in Sethuraman and Vemuri [128] used a force-directed scheduling algorithm to optimise bandwidth in NoC (Networks-on-Chip) architecture by scheduling an optimal size (dimension mesh) of the network circuit.

3.3 FORCE-DIRECTED ALGORITHMS IN INFORMATION VISUALISATION

The primary objective of network information visualisation is to explore hidden patterns in networks and to visualise them in a simple manner. Information networks can be social networks, human relation networks, networks of business workflow, transportation maps, etc. An example visualisation of a clustered network is illustrated in Figure 6 (b) which is generated by using the vis.js visualisation tool [129]. The application of information visualisation is wide and complex. It is therefore impossible to visualise networks in orthogonal form or in a planar visual drawing in all cases. Moreover, certain information networks' nodes and edges may contain additional properties (attributes). These properties do not exist in general networks. Because of this, applications of information visualisation may use variant layouts to present the data. For example, metro map diagramming is useful for visualising the transportation map as a schematic [40]. With fisheye views [130, 131], the network representation can enlarge regions located near specified nodes while contracting distant regions by varying edge length. However, while enlarging a special region may be useful in two-dimensional planes, it may not be applicable for high-dimensional data. Parallel coordinate diagramming can be used to project high-dimensional data onto two dimensions [132]. Parallel coordinate diagrams draw n vertical lines equally spaced to represent the n-dimensional space. Corresponding nodes are drawn on the dimensional space (vertical line) and the line represents the relation between a pair of nodes [133]. Lombardi-Style diagrams [134] are useful for information visualisation in which the edges of the visual drawing are curvilinear [135].

Even with these techniques, the amount of complexity makes visual interpretability for humans difficult. Chae [136] suggested visualising large networks on a tiled monitor wall, in which monitors are placed next to each other and data distribution related to their corresponding display nodes are only displayed. Edge crossing deduction is also crucial for visualising information about large networks, as this makes the representation appear cluttered and ugly. One example is the 1/4-SHPED (i.e. Symmetric Homogeneous Partial Edge Drawing), as proposed by Bruckdorfer et al. [137]. Nodes of 1/4-SHPED are represented as points and edges as two pieces (also called stubs) of a straight-line segment, each adjacent to a node, without any edge crossings, and with stub size 1/4 of the total edge length. Edge bundling is another technique which group edges into bundles to decrease the density of lines for reducing clutter [131-133]. Moreover, Debiasi et al. [33] proposed an accumulated force model to visualise the network by geographical flow map [33] as a way to prevent edge crossings. Each flow consists of start, intermediate and target nodes and three forces are defined in the proposed algorithm: electrostatic (attractive) force, stress force and rejected (repulsive) force. The force equation of node v is the sum of the three forces and can be defined as follows:

$$F(v) = \sum_{s \in S} F_e(v, s) + \sum_{t \in T} F_r(v, t) + F_s(v) \tag{31}$$

where S is the intermediate nodes interacting with node v. T is the nodes near to the node v. The purpose of electrostatic force (F_e) is similar to attractive force, which is defined as follows:

$$F_e(v, s) = \frac{1}{\|v - s\|} \times \widehat{v - s} \tag{32}$$

where $\widehat{v - s}$ is the unit vector of node v and s. $\|v - s\|$ is the norm of node v and s. Stress force enables the node to move towards the flow with higher magnitude. The definition of stress force is as follows:

$$F_s(v) = (v_{i-1} - v) + (v - v_{i+1}) \tag{33}$$

where v_{i+1} is the ancestor node of v and v_{i+1} is the child node of v. Rejected forces are used to avoid any overlapping between intermediate and target nodes, and are defined as follows:

$$F_r(v,t) = -1 \times F_e(v,s) \quad (34)$$

There are approaches which utilize forces and reduce clutter for graph visualisation in which the size of nodes in the graph is the variant. Cui et al. [134] used a force-directed model to visualise word clouds in which the size of a word (i.e. each word represents a node) is determined by the word frequency in the time slot. Moreover, Gu et al. [135] adopted the FR algorithm to visual large texts and image datasets on a large video wall. We summarise the relevant studies on information visualisation, their objectives, and corresponding force-directed algorithms used to visualise the networks in Table 2.

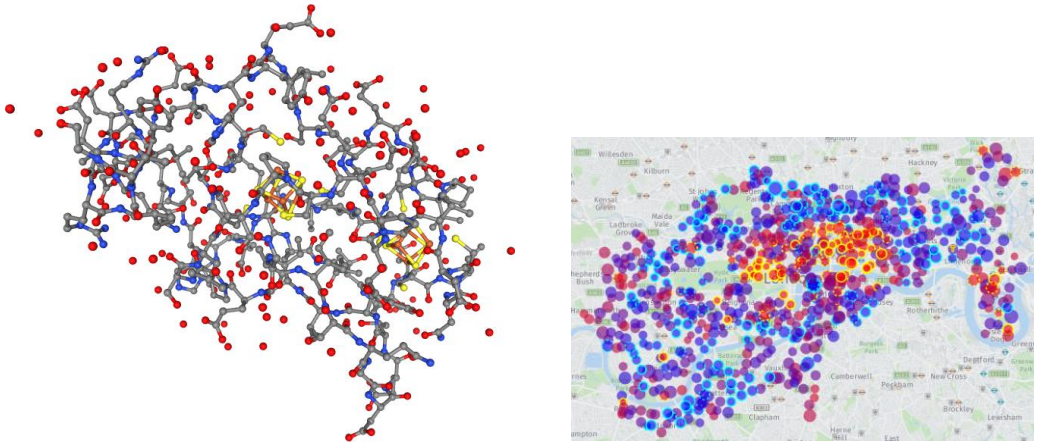
Table 2 Information visualisation studies that adopted force-directed algorithms.

Force-directed algorithm adopted by the study	**Study**	**Objective of the study**
Eades's algorithm	[1]	− To visualise networks using tree-structured hierarchies − Increase the readability of a network
	[2]	− To make edges conform to particular orientations
	[3]	− To transform the Extensible Stylesheet Language Transformations (XSLT) document to network layout. XSLT is a language for transforming vector images and documents in the XML encoding
	[4]	− To visualise networks with non-uniform nodes (i.e. the size and shape of nodes are variant)
	[5]	− To visualise the relation of people in online social networks
	[6]	− To visualise networks using grid layouts
	[7]	− To visualise web traffic
FR algorithm	[8]	− To visualise networks in which nodes have nontrivial sizes
	[9]	− To produce visual drawings of hypergraphs − Hypergraphs can be viewed as an extension of classical networks in which an edge can join any number of vertices
	[10]	− To visualise networks with non-uniform nodes (i.e. the size and shape of nodes are variant)
	[11]	− To visualise exploration of network traffic over time
	[12]	− To visualise email networks
	[13]	− To visualise transportation networks
	[14]	− To visualise networks in which edges are curvilinear (Bézier curve)
	[15]	− To visualise weighted networks in which each edge is associated with a real number representing its importance
	[16]	− To assess homophily [17] in networks
	[18]	− To visualise the actors holding neutral opinion polarities
	[19]	− To visualise the volume of movement in flow maps
KK algorithm	[20]	− To show proximity between nodes such that their distances in the visualisation reflect distances in the network − topology
	[21]	− To visualise networks in which edges are curvilinear
	[22]	− To visualise the structure of ER diagram
	[23]	− To visualise the count of paper submissions for journal articles of natural and social sciences
	[24]	− To animate networks over time

Noack algorithm [25]	[26]	− To visualise the land and water networks of port transportation
FA2 algorithm [27]	[28]	− To visualise the transaction patterns of Bitcoin networks
Hachul algorithm [29]	[30]	− To use k-dimensional trese (a data structure of space partitioning for arranging nodes in a k-dimensional space) to visualise networks.

3.4 FORCE-DIRECTED ALGORITHMS IN BIOLOGICAL NETWORK VISUALISATION

Visualisation is an important way to capture the dependencies and interactions between different biological entities, and their sequential processes. The force-directed algorithm is one of the most popular approaches for the visualisation of biological networks. An example visualisation of a biological network is illustrated in Figure 7 (a) (which is generated by using the NGL molecular visualisation viewer [167]). Kerpedjiev et al. [168] developed a tool called $forna$ to display the secondary structure of ribonucleic acid (RNA). In addition, Bang et al. [169] and Tuikkala et al. [170] proposed multilevel force-directed algorithms to visualise large protein networks and genetic interactions.



(a) (b)

Figure 7 An example visualisation of (a) a biological network, (b) sensor localisation.

Biological networks have more special attributes than average directed and undirected networks. Because of this, various researchers have proposed special layouts for the visualisation of genetic sequencing or other biological networks. Clustered layouts are commonly used to visualise protein interactions [171]. Gamma-Clustering layouts were suggested for visualising large and complex biological networks [172]. Haplotype layouts [173] were also used to distinguish relationships among different sequences observed in biological networks [174]. There are also several studies that adopt force-directed algorithms to visualise the structure of molecules, biological pathways, protein networks, etc. Due to the page limitation, we summarise these studies in Table 3.

Table 3 Visualisation studies that adopted force-directed algorithms for biological networks.

Force-directed algorithm adopted by the study	Study	Objective of the study
KK algorithm	[175]	− To visualise protein–protein interaction network
	[178]	− To visualise protein–protein interaction network
	[185]	− To visualise the structure of Alpha-helical transmembrane proteins

FR algorithm	[176]	− To use Schlegel diagrams [177] to visualise the structure of molecules
	[179], [180], [181]	− To visualise biological pathways
	[182]	− To visualise the structure of genes − Minimise edge-edge crossings
	[183]	− To visualise the structure of genes
	[184]	− To visualise microarrays
	[186]	− To visualise biological pathways
	[187]	− To analyse the connectivity patterns of brain parcellation
	[172]	− To visualise large biological networks

3.5 FORCE-DIRECTED ALGORITHMS IN NODE PLACEMENT AND LOCALISATION FOR SENSOR NETWORKS

Sensor networks are useful for monitoring animals, earthquakes and tsunamis [188], emergency message forwarding during disasters [189], etc. An example visualisation of sensor localization is illustrated in Figure 5 (b) (which is generated by using the OOMap service [190]). Because the exact location of the networked sensors (nodes) is often unavailable, force-directed algorithms are used to determine node placement or to locate boundaries to improve the network's coverage [191]. The strength of force is subject to the distance between two nodes and each node behaves as a source of force. Therefore, if the distance between two nodes is shorter/larger than a threshold, a repulsive/attractive force will be exerted on each other. If the distance is equal to the threshold, no force will act upon the nodes. Extensions of the FR algorithm were proposed for node placement in [192, 193]. There are also extensions based on a modified FR algorithm that estimate the approximate location of each node based on signal information [194, 195]. In [196], Cheong and Si proposed a heuristic KK algorithm for boundary detection. The proposed algorithm was optimised for sending emergency messages via Mobile Ad Hoc network if cellular networks are corrupted. Nodes on the boundary are responsible for forwarding emergency messages to nearly emergency stations.

4 CONCLUSIONS

In this paper, we present the survey of force-directed algorithms for schematic drawings and placement. This class of algorithms has been studied and implemented in biological network visualisation, information visualisation, sensor localisation and VLSI design. This survey covers classical force-directed algorithms and hybrid force-directed algorithms, in which parallel, multilevel and multidimensional scaling techniques are used. We also discussed the merits and deficiencies of force-directed algorithms and visualisation applications. For example, how network topologies are drawn can significantly affect viewers' understanding of the network. We also discussed the influences caused by the layout and position-assignment of visualised network nodes on how a user perceives the relationships in the network. To this end, we review and categorise force-directed algorithms from research areas such as: (a) aesthetic drawings for general networks, (b) component placement and scheduling in high-level synthesis of very-large scale integration (VLSI) circuits design, (c) information visualisation, (d) biological network visualisation, and (e) node placement and localisation for sensor networks. Our hope is that this survey not only provides an overview of existing force-directed algorithms, but also introduces them as effective tools for solving visualisation problems in different application areas.

REFERENCES

[1] J. S. Richardson, "Schematic drawings of protein structures," in *Methods in enzymology*, vol. 115: Elsevier, 1985, pp. 359-380.

[2] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice & Experience*, vol. 21, no. 11, pp. 1129 - 1164, 1991.

[3] I. ACM. (2018, September 1). *ACM Digital Library*. Available: https://dl.acm.org

[4] E. B.V. (2018, September 1). *Scopus Preview*. Available: https://www.scopus.com

[5] H. Gibson, J. Faith, and P. Vickers, "A survey of two-dimensional graph layout techniques for information visualisation," *Information visualization*, vol. 12, no. 3-4, pp. 324-357, 2013.

[6] S. G. Kobourov, "Spring embedders and force directed graph drawing algorithms," *arXiv preprint arXiv:1201.3011*, 2012.

[7] R. Tamassia, *Handbook of graph drawing and visualization*. CRC press, 2013.

[8] P. Eades and R. Tamassia, *Algorithms for drawing graphs: an annotated bibliography*. Department of Computer Science, Brown University Providence, 1989.

[9] F. J. Brandenburg, M. Himsolt, and C. Rohrer, "An experimental comparison of force-directed and randomized graph drawing algorithms," in *International Symposium on Graph Drawing*, 1995, pp. 76-87: Springer.

[10] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, "Algorithms for drawing graphs: an annotated bibliography," *Computational Geometry*, vol. 4, no. 5, pp. 235-282, 1994.

[11] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall PTR, 1998.

[12] M. Himsolt, "GraphEd: A graphical platform for the implementation of graph algorithms (extended abstract and demo)," in *Graph Drawing*, 1995, pp. 182-193: Springer.

[13] D. Dodson, "COMAIDE: Information visualization using cooperative 3D diagram layout," in *International Symposium on Graph Drawing*, 1995, pp. 190-201: Springer.

[14] L. Behzadi, "LayoutShow: A signed applet/application for graph drawing and experimentation," in *Graph Drawing*, 1999, pp. 242-249: Springer.

[15] D. Forrester, S. G. Kobourov, A. Navabi, K. Wampler, and G. V. Yee, "Graphael: A System for Generalized Force-Directed Layouts," in *Graph drawing*, 2004, pp. 454-464: Springer.

[16] S. Martin, W. M. Brown, R. Klavans, and K. W. Boyack, "OpenOrd: An open-source toolbox for large graph layout," in *Visualization and Data Analysis*, 2011, p. 786806.

[17] J. Ignatowicz, "Drawing force-directed graphs using Optigraph," in *International Symposium on Graph Drawing*, 1995, pp. 333-336: Springer.

[18] C. Bennett, J. Ryall, L. Spalteholz, and A. Gooch, "The aesthetics of graph visualization," *Computational aesthetics*, vol. 2007, pp. 57-64, 2007.

[19] H. C. Purchase, "Effective information visualisation: a study of graph drawing aesthetics and algorithms," *Interacting with computers*, vol. 13, no. 2, pp. 147-162, 2000.

[20] M. Huang, W. Huang, and C.-C. Lin, "Evaluating force-directed algorithms with a new framework," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 1030-1032: ACM.

[21] D. Tunkelang, "A practical approach to drawing undirected graphs," DTIC Document1994.

[22] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software," *PLoS ONE*, vol. 9, no. 6, 2014.

[23] Q.-G. Zhang, H.-Y. Liu, W. Zhang, and Y.-J. Guo, "Drawing undirected graphs with genetic algorithms," *Advances in Natural Computation*, pp. 435-435, 2005.

[24] U. Dogrusoz, E. Giral, A. Cetintas, A. Civril, and E. Demir, "A layout algorithm for undirected compound graphs," *Information Sciences*, vol. 179, no. 7, pp. 980-994, 2009.

[25] M. J. Bannister, D. Eppstein, M. T. Goodrich, and L. Trott, "Force-directed graph drawing using social gravity and scaling," in *Graph Drawing*, 2013, vol. 7704, pp. 414-425.

[26] P. Eades, "A heuristic for graph drawing," *Congressus numerantium*, vol. 42, pp. 149 - 160, 1984.

[27] A. Frick, A. Ludwig, and H. Mehldau, "A fast adaptive layout algorithm for undirected graphs (extended abstract and system demonstration)," in *Graph Drawing*, 1995, pp. 388-403: Springer.

[28] W. Huang, P. Eades, S.-H. Hong, and C.-C. Lin, "Improving multiple aesthetics produces better graph drawings," *Journal of Visual Languages & Computing*, vol. 24, no. 4, pp. 262-272, 2013.

[29] R. Davidson and D. Harel, "Drawing graphs nicely using simulated annealing," *ACM Transactions on Graphics*, vol. 15, no. 4, pp. 301 - 331, 1996.

[30] D. Harel and Y. Koren, "A fast multi-scale method for drawing large graphs," *Journal of graph algorithms and applications*, vol. 6, no. 3, pp. 179 - 202, 2001.

[31] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7 - 15, 1989.
[32] A. Noack, "Energy Models for Graph Clustering," *J. Graph Algorithms Appl.*, vol. 11, no. 2, pp. 453-480, 2007.
[33] A. Debiasi, B. Simões, and R. De Amicis, "Force directed flow map layout," in *Information Visualization Theory and Applications (IVAPP), 2014 International Conference on*, 2014, pp. 170-177: IEEE.
[34] F. Bertault, "A force-directed algorithm that preserves edge crossing properties," in *Graph Drawing*, 1999, pp. 351-358: Springer.
[35] W. Didimo, G. Liotta, and S. A. Romeo, "Topology-Driven Force-Directed Algorithms," in *Graph drawing*, 2010, vol. 6502, pp. 165-176: Springer.
[36] W. Dong, X. Fu, G. Xu, and Y. Huang, "An improved force-directed graph layout algorithm based on aesthetic criteria," *Computing and Visualization in Science*, vol. 16, no. 3, pp. 139-149, 2013.
[37] C. Papadopoulos and C. Voglis, "Untangling graphs representing spatial relationships driven by drawing aesthetics," in *Proceedings of the 17th Panhellenic Conference on Informatics*, 2013, pp. 158-165: ACM.
[38] P. Eades, W. Huang, and S.-H. Hong, "A force-directed method for large crossing angle graph drawing," *arXiv preprint arXiv:1012.4559*, 2010.
[39] E. N. Argyriou, M. A. Bekos, and A. Symvonis, "Maximizing the Total Resolution of Graphs," in *Graph Drawing*, 2010, pp. 62-67: Springer.
[40] D. Chivers and P. Rodgers, "Octilinear force-directed layout with mental map preservation for schematic diagrams," in *International Conference on Theory and Application of Diagrams*, 2014, pp. 1-8: Springer.
[41] K. Sugiyama and K. Misue, "A simple and unified method for drawing graphs: Magnetic-spring algorithm," in *International Symposium on Graph Drawing*, 1994, pp. 364-375: Springer.
[42] G. James, B. Silverman, and B. Silverman. (2018, September 6). *Visual Transistor-level Simulation*. Available: http://www.visual6502.org
[43] N. R. Quinn Jr, "The placement problem as viewed from the physics of classical mechanics," in *Papers on Twenty-five years of electronic design automation*, 1988, pp. 67-72: ACM.
[44] D. C. Wilson and R. J. Smith II, "An experimental comparison of force directed placement techniques," in *Proceedings of the 11th Design Automation Workshop*, 1974, pp. 194-199: IEEE Press.
[45] C. J. Fisk and D. Isett, "'"ACCEL" automated circuit card etching layout," in *Proceedings of the SHARE design automation project*, 1965, pp. 9.1-9.31: ACM.
[46] N. Quinn and M. Breuer, "A forced directed component placement procedure for printed circuit boards," *IEEE Transactions on Circuits and systems*, vol. 26, no. 6, pp. 377-388, 1979.
[47] S. J. Urban, N. C. Randall, and T. J. Harley, "System for Heuristic and Rapid Processing of Component Layout and Wiring (SHARPCLAW)," in *Proceedings of the 1971 26th annual conference*, 1971, pp. 414-432: ACM.
[48] G. Wipfler, M. Wiesel, and D. A. Mlynski, "A combined force and cut algorithm for hierarchical VLSI layout," in *Proceedings of the 19th Design Automation Conference*, 1982, pp. 671-677: IEEE Press.
[49] M. Hanan, P. K. Wolff, and B. J. Agule, "Some experimental results on placement techniques," in *Papers on Twenty-five years of electronic design automation*, 1988, pp. 73-83: ACM.
[50] F. Johannes, K. M. Just, and K. J. Antreich, "On the force placement of logic arrays," in *Proceedings of the 6th European Conference on Cmcuit Theory and Design. pp*, 1983, pp. 203-206.
[51] K. Antreich, F. Johannes, and F. Kirsch, "A new approach for solving the placement problem using force models," in *Proceedings of the IEEE International Symposmm on C%-cuits and Systems. pp*, 1982, pp. 481-486.
[52] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 3, pp. 356-365, 1991.
[53] K. Shahookar and P. Mazumder, "VLSI cell placement techniques," *ACM Computing Surveys (CSUR)*, vol. 23, no. 2, pp. 143-220, 1991.
[54] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proceedings of the 35th annual Design Automation Conference*, 1998, pp. 269-274: ACM.
[55] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2—A fast force-directed quadratic placement approach using an accurate net model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1398-1411, 2008.
[56] B. Hu and M. Marek-Sadowska, "FAR: Fixed-points addition & relaxation based placement," in *Proceedings of the 2002 international symposium on Physical design*, 2002, pp. 161-166: ACM.
[57] B. Hu and M. Marek-Sadowska, "Multilevel fixed-point-addition-based VLSI placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 8, pp. 1188-1203, 2005.

[58] A. Kennings and K. P. Vorwerk, "Force-directed methods for generic placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2076-2087, 2006.
[59] K. Vorwerk, A. Kennings, and A. Vannelli, "Engineering details of a stable force-directed placer," in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, 2004, pp. 573-580: IEEE Computer Society.
[60] N. Viswanathan and C.-N. Chu, "FastPlace: efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 5, pp. 722-733, 2005.
[61] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *Proceedings of the 2007 Asia and South Pacific Design Automation Conference*, 2007, pp. 135-140: IEEE Computer Society.
[62] N. Viswanathan, G.-J. Nam, C. J. Alpert, P. Villarrubia, H. Ren, and C. Chu, "RQL: Global placement via relaxed quadratic spreading and linearization," in *Proceedings of the 44th annual Design Automation Conference*, 2007, pp. 453-458: ACM.
[63] M.-C. Kim, D.-J. Lee, and I. L. Markov, "SimPL: An effective placement algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 1, pp. 50-60, 2012.
[64] H. Eisenmann, "Force-Directed Placement of VLSI Circuits," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, pp. 131-132: ACM.
[65] B. Obermeier, H. Ranke, and F. M. Johannes, "Kraftwerk: a versatile placement approach," in *Proceedings of the 2005 international symposium on Physical design*, 2005, pp. 242-244: ACM.
[66] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proceedings of the 2005 international symposium on Physical design*, 2005, pp. 185-192: ACM.
[67] T. F. Chan, J. Cong, M. Romesis, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: A robust multilevel mixed-size placement engine," in *Proceedings of the 2005 international symposium on Physical design*, 2005, pp. 227-229: ACM.
[68] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: enhanced multilevel mixed-size placement," in *Proceedings of the 2006 international symposium on Physical design*, 2006, pp. 212-214: ACM.
[69] S. W. Hur and J. Lillis, "Mongrel: hybrid techniques for standard cell placement," in *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, 2000, pp. 165-170: IEEE Press.
[70] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *IEEE Transactions on Circuits and Systems*, vol. 28, no. 1, pp. 12-18, 1981.
[71] S. Goto and T. Matsuda, "Partitioning, assignment and placement," *Layout Design And Verification,'T. Ohtsuki, Ed. Elsevier North-Holland, New York*, pp. 55-97, 1986.
[72] Y. W. Chang, "Generalized Force Directed Relaxation with Optimal Regions and Its Applications to Circuit Placement," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, 2017, pp. 115-120: ACM.
[73] G. Odawara, K. Iijima, and K. Wakabayashi, "Knowledge-based placement technique for printed wiring boards," in *Proceedings of the 22nd ACM/IEEE Design Automation Conference*, 1985, pp. 616-622: IEEE Press.
[74] S. Alupoaei and S. Katkoori, "Net clustering based macrocell placement," in *Proceedings of the 2002 Asia and South Pacific Design Automation Conference*, 2002, p. 399: IEEE Computer Society.
[75] C.-J. Tseng and D. P. Siewiorek, "Facet: A procedure for the automated synthesis of digital systems," in *Proceedings of the 20th Design Automation Conference*, 1983, pp. 490-496: IEEE Press.
[76] K. Vorwerk and A. Kennings, "An improved multi-level framework for force-directed placement," in *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, 2005, pp. 902-907: IEEE Computer Society.
[77] J. J. Cong and J. R. Shinnerl, *Multilevel optimization in VLSICAD*. Springer Science & Business Media, 2013.
[78] Y.-C. Chou and Y.-L. Lin, "A performance-driven standard-cell placer based on a modified force-directed algorithm," in *Proceedings of the 2001 international symposium on Physical design*, 2001, pp. 24-29: ACM.
[79] M.-C. Kim, J. Hu, D.-J. Lee, and I. L. Markov, "A SimPLR method for routability-driven placement," in *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, 2011, pp. 67-73: IEEE.
[80] M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov, and S. Ramji, "MAPLE: multilevel adaptive placement for mixed-size designs," in *Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design*, 2012, pp. 193-200: ACM.

[81] U. Brenner, M. Struzyna, and J. Vygen, "BonnPlace: Placement of leading-edge chips by advanced combinatorial algorithms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1607-1620, 2008.

[82] U. Brenner, A. Hermann, N. Hoppmann, and P. Ochsendorf, "BonnPlace: A self-stabilizing placement framework," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, pp. 9-16: ACM.

[83] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1228-1240, 2008.

[84] R. Forbes, "Heuristic acceleration of force-directed placement," in *Proceedings of the 24th ACM/IEEE Design Automation Conference*, 1987, pp. 735-740: ACM.

[85] P. Spindler and F. M. Johannes, "Fast and robust quadratic placement combined with an exact linear net model," in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, 2006, pp. 179-186: ACM.

[86] G.-J. Nam and J. J. Cong, *Modern circuit placement: best practices and results*. Springer Science & Business Media, 2007.

[87] S.-W. Hur *et al.*, "Force directed mongrel with physical net constraints," in *Proceedings of the 40th annual Design Automation Conference*, 2003, pp. 214-219: ACM.

[88] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, 2003, p. 86: IEEE Computer Society.

[89] D. R. Brasen and M. L. Bushnell, "MHERTZ: A new optimization algorithm for floorplanning and global routing," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, 1991, pp. 107-110: ACM.

[90] H. Youssef, S. M. Sait, and K. J. Al-Farra, "Timing influenced force directed floorplanning," in *Proceedings of the conference on European design automation*, 1995, pp. 156-161: IEEE Computer Society Press.

[91] M. H. Arnold and W. S. Scott, "An interactive maze router with hints," in *Proceedings of the 25th ACM/IEEE Design Automation Conference*, 1988, pp. 672-676: IEEE Computer Society Press.

[92] F. Mo, A. Tabbara, and R. K. Brayton, "A force-directed maze router," in *Computer Aided Design, 2001. ICCAD 2001. IEEE/ACM International Conference on*, 2001, pp. 404-407: IEEE.

[93] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary, and B. Halpin, "Timing driven force directed placement with physical net constraints," in *Proceedings of the 2003 international symposium on Physical design*, 2003, pp. 60-66: ACM.

[94] P. Saxena and B. Halpin, "Modeling repeaters explicitly within analytical placement," in *Proceedings of the 41st annual Design Automation Conference*, 2004, pp. 699-704: ACM.

[95] Y. I. Ismail and E. G. Friedman, "Effects of inductance on the propagation delay and repeater insertion in VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 2, pp. 195-206, 2000.

[96] B. Goplen, P. Saxena, and S. Sapatnekar, "Net weighting to reduce repeater counts during placement," in *Proceedings of the 42nd annual Design Automation Conference*, 2005, pp. 503-508: ACM.

[97] J. Cong, G. Luo, and E. Radke, "Highly efficient gradient computation for density-constrained analytical placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 12, pp. 2133-2144, 2008.

[98] H. Ren, D. Z. Pan, C. J. Alpert, and P. Villarrubia, "Diffusion-based placement migration," in *Proceedings of the 42nd annual Design Automation Conference*, 2005, pp. 515-520: ACM.

[99] K. Tsota, C.-K. Koh, and V. Balakrishnan, "A Mixed-size Placer based on Dimension Adjustment."

[100] G. Sigl, K. Doll, and F. M. Johannes, "Analytical placement: A linear or a quadratic objective function?," in *Proceedings of the 28th ACM/IEEE Design Automation Conference*, 1991, pp. 427-432: ACM.

[101] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-programmable gate arrays*. Springer Science & Business Media, 2012.

[102] C. Ababei *et al.*, "Placement and routing in 3D integrated circuits," *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 520-531, 2005.

[103] W. Sui, S. Dong, and J. Bian, "Wirelength-driven force-directed 3D FPGA placement," in *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, 2010, pp. 435-440: ACM.

[104] S. Kumar *et al.*, "A network on chip architecture and design methodology," in *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, 2002, pp. 117-124: IEEE.

[105] A. von Beuningen and U. Schlichtmann, "Platon: A force-directed placement algorithm for 3d optical networks-on-chip," in *Proceedings of the 2016 on International Symposium on Physical Design*, 2016, pp. 27-34: ACM.

[106] L. Stock and R. van der Born, "EASY: multiprocessor architecture optimization," in *Workshop logic and architecture synthesis for silicon compilers*, 1988.
[107] W. Verhaegh, "Scheduling problems in video signal processing," *Master's thesis. Eindhoven Univ. of Technol., Eindhoven, The Netherlands*, 1990.
[108] R. Camposano and W. Wolf, *High-level VLSI synthesis*. Springer Science & Business Media, 2012.
[109] G. Wang, W. Gong, and R. Kastner, "Instruction scheduling using MAX-MIN ant system optimization," in *Proceedings of the 15th ACM Great Lakes symposium on VLSI*, 2005, pp. 44-49: ACM.
[110] P. G. Paulin and J. P. Knight, "Force-directed scheduling in automatic data path synthesis," in *Proceedings of the 24th ACM/IEEE Design Automation Conference*, 1987, pp. 195-202: ACM.
[111] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 6, pp. 661-679, 1989.
[112] P. G. Paulin and J. P. Knight, "Scheduling and binding algorithms for high-level synthesis," in *Design Automation, 1989. 26th Conference on*, 1989, pp. 1-6: IEEE.
[113] R. J. Cloutier and D. E. Thomas, "The combination of scheduling, allocation, and mapping in a single algorithm," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, 1991, pp. 71-76: ACM.
[114] C. Lee, M. Potkonjak, and W. Wolf, "System-level synthesis of application specific systems using A* search and generalized force-directed heuristics," in *Proceedings of the 9th international symposium on System synthesis*, 1996, p. 2: IEEE Computer Society.
[115] R. F. Abdel-Kader, "Resource-constrained loop scheduling in high-level synthesis," in *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, 2005, pp. 195-200: ACM.
[116] W. Zhang, L. Shang, and N. K. Jha, "NanoMap: An integrated design optimization flow for a hybrid nanotube/CMOS dynamically reconfigurable architecture," in *Proceedings of the 44th annual Design Automation Conference*, 2007, pp. 300-305: ACM.
[117] D. Chang and M. Marek-Sadowska, "Partitioning sequential circuits on dynamically reconfigurable FPGAs," *IEEE Transactions on Computers*, vol. 48, no. 6, pp. 565-578, 1999.
[118] T. J.-F. Omnés, T. Franzetti, and F. Catthoor, "Interactive co-design of high throughput embedded multimedia," in *Proceedings of the 37th Annual Design Automation Conference*, 2000, pp. 328-331: ACM.
[119] S. Katkoori and R. Vemuri, "Scheduling for low power under resource and latency constraints," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, 2000, vol. 2, pp. 53-56: IEEE.
[120] S. Gupta and S. Katkoori, "Force-directed scheduling for dynamic power optimization," in *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, 2002, pp. 75-80: IEEE.
[121] A. Allam and J. Ramanujam, "Modified force-directed scheduling for peak and average power optimization using multiple supply-voltages," in *Integrated Circuit Design and Technology, 2006. ICICDT'06. 2006 IEEE International Conference on*, 2006, pp. 1-5: IEEE.
[122] T. Schönwald, A. Viehl, O. Bringmann, and W. Rosenstiel, "Optimized software mapping for advanced driver assistance systems," in *Industrial Embedded Systems (SIES), 2012 7th IEEE International Symposium on*, 2012, pp. 181-190: IEEE.
[123] W. F. Verhaegh, E. H. Aarts, J. H. Korst, and P. E. Lippens, "Improved force-directed scheduling," in *Design Automation. EDAC., Proceedings of the European Conference on*, 1991, pp. 430-435: IEEE.
[124] P. E. Lippens *et al.*, "PHIDEO: a silicon compiler for high speed algorithms," in *Proceedings of the conference on European design automation*, 1991, pp. 436-441: IEEE Computer Society Press.
[125] W. F. Verhaegh, P. E. Lippens, E. H. Aarts, J. H. Korst, A. van der Werf, and J. L. van Meerbergen, "Efficiency improvements for force-directed scheduling," in *Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design*, 1992, pp. 286-291: IEEE Computer Society Press.
[126] W. Zhang, N. K. Jha, and L. Shang, "NATURE: A hybrid nanotube/CMOS dynamically reconfigurable architecture," in *Design Automation Conference, 2006 43rd ACM/IEEE*, 2006, pp. 711-716: IEEE.
[127] T. Schönwald, A. Viehl, O. Bringmann, and W. Rosenstiel, "Shared memory aware MPSoC software deployment," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2013, pp. 1771-1776: EDA Consortium.
[128] B. Sethuraman and R. Vemuri, "A force-directed approach for fast generation of efficient multi-port NoC architectures," in *VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*, 2007, pp. 419-426: IEEE.
[129] A. B.V. (2017, September 9). *vis.js*. Available: http://www.visjs.org

[130] J. Nagumo and J. Tanaka, "Introducing the fisheye view into graph drawing algorithm," *Systems and Computers in Japan*, vol. 32, no. 12, pp. 60-66, 2001.
[131] G. W. Furnas, "The FISHEYE view: A new look at structured files," Bell Laboratories Technical Memorandum1981.
[132] R. Walker, P. A. Legg, S. Pop, Z. Geng, R. S. Laramee, and J. C. Roberts, "Force-directed parallel coordinates," in *Information Visualisation (IV), 2013 17th International Conference*, 2013, pp. 36-44: IEEE.
[133] A. Inselberg and B. Dimsdale, "Parallel coordinates: a tool for visualizing multi-dimensional geometry.(1990)," *DOI: http://dx. doi. org/10.1109/VISUAL*, 1990.
[134] M. Lombardi and R. C. Hobbs, *Mark Lombardi: Global Networks*. Independent Curators, 2003.
[135] R. Chernobelskiy, K. I. Cunningham, M. T. Goodrich, S. G. Kobourov, and L. Trott, "Force-directed lombardi-style graph drawing," in *Graph Drawing*, 2011, vol. 7034, pp. 320-331: Springer.
[136] S. Chae, "Distributed graph visualization on tiled displays," in *Proc. of SPIE Vol*, 2013, vol. 8768, pp. 87687K-1.
[137] T. Bruckdorfer, M. Kaufmann, and A. Lauer, "A practical approach for 1/4-shpeds," in *Information, Intelligence, Systems and Applications (IISA), 2015 6th International Conference on*, 2015, pp. 1-6: IEEE.
[138] J. Barnes and P. Hut, "A hierarchical O (N log N) force-calculation algorithm," *nature*, vol. 324, no. 6096, pp. 446-449, 1986.
[139] K. Sugiyama and K. Misue, "Graph drawing by the magnetic spring model," *Journal of Visual Languages & Computing*, vol. 6, no. 3, pp. 217-231, 1995.
[140] X. Wang and I. Miyamoto, "Generating customized layouts," in *Graph Drawing*, 1996, pp. 504-515: Springer.
[141] J. D. Cohen, "Drawing graphs to convey proximity: An incremental arrangement method," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 4, no. 3, pp. 197-229, 1997.
[142] F. Bertault and P. Eades, "Drawing hypergraphs in the subset standard (short demo paper)," in *Graph Drawing*, 2001, pp. 45-76: Springer.
[143] D. Harel and Y. Koren, "Drawing graphs with non-uniform vertices," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, 2002, pp. 157-166: ACM.
[144] J.-H. Chuang, C.-C. Lin, and H.-C. Yen, "Drawing graphs with nonuniform nodes using potential fields," in *Graph Drawing*, 2003, vol. 2912, pp. 460-465: Springer.
[145] N. Churcher, W. Irwin, and C. Cook, "Inhomogeneous force-directed layout algorithms in the visualisation pipeline: From layouts to visualisations," in *Proceedings of the 2004 Australasian symposium on Information Visualisation-Volume 35*, 2004, pp. 43-51: Australian Computer Society, Inc.
[146] B. Finkel and R. Tamassia, "Curvilinear Graph Drawing Using the Force-Directed Method," in *Graph Drawing*, 2004, vol. 3383, pp. 448-453: Springer.
[147] Y. Tzitzikas and J.-L. Hainaut, "How to tame a very large ER diagram (using link analysis and force-directed drawing algorithms)," *ER*, vol. 3716, pp. 144-159, 2005.
[148] K. W. Boyack, R. Klavans, and K. Börner, "Mapping the backbone of science," *Scientometrics*, vol. 64, no. 3, pp. 351-374, 2005.
[149] J. Heer and D. Boyd, "Vizster: Visualizing online social networks," in *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, 2005, pp. 32-39: IEEE.
[150] S. Bender-deMoll and D. A. McFarland, "The art and science of dynamic network visualization," *Journal of Social Structure*, vol. 7, no. 2, pp. 1-38, 2006.
[151] U. Lauther, "Multipole-based force approximation revisited–a simple but fast implementation using a dynamized enclosing-circle-enhanced kd-tree," in *International Symposium on Graph Drawing*, 2006, pp. 20-29: Springer.
[152] S. Hachul, "A Potential-Field-Based Multilevel Algorithm for Drawing Large Graphs," Universität zu Köln, 2002.
[153] F. Mansman, L. Meier, and D. A. Keim, "Visualization of host behavior for network security," in *VizSEC 2007*: Springer, 2008, pp. 187-202.
[154] X. Wang, X. Zhou, W. Lana, and S. Wu, "An improved graph drawing algorithm for email networks," in *Asian Control Conference, 2009. ASCC 2009. 7th*, 2009, pp. 1667-1672: IEEE.
[155] U. Brandes, G. Shubina, R. Tamassia, and D. Wagner, "Fast layout methods for timetable graphs," in *International Symposium on Graph Drawing*, 2000, pp. 127-138: Springer.
[156] W. Didimo, G. Liotta, and S. A. Romeo, "A Graph Drawing Application to Web Site Traffic Analysis," *J. Graph Algorithms Appl.*, vol. 15, no. 2, pp. 229-251, 2011.
[157] M. Fink, H. Haverkort, M. Nöllenburg, M. Roberts, J. Schuhmann, and A. Wolff, "Drawing metro maps using Bézier curves," in *International Symposium on Graph Drawing*, 2012, pp. 463-474: Springer.
[158] S. Kieffer, T. Dwyer, K. Marriott, and M. Wybrow, "Incremental Grid-Like Layout Using Soft and Hard Constraints," in *Graph Drawing*, 2013, pp. 448-459.

[159] J. Hua, M. L. Huang, and Q. V. Nguyen, "Drawing large weighted graphs using clustered force-directed algorithm," in *Information Visualisation (IV), 2014 18th International Conference on*, 2014, pp. 13-17: IEEE.
[160] W. Meulemans and A. Schulz, "A Tale of Two Communities: Assessing Homophily in Node-Link Diagrams," in *International Symposium on Graph Drawing and Network Visualization*, 2015, pp. 489-501: Springer.
[161] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415-444, 2001.
[162] X. Du, Y. Ye, Y. Raymond, K. Lau, Y. Li, and X. Huang, "Multi-opinion Ring: visualizing and predicting multiple opinion orientations in online social media," *Multimedia Tools and Applications*, vol. 75, no. 12, p. 7159, 2016.
[163] G. Yan and L. Huang, "Research on Visualization of Port Transportation Network based on Force Directed Model," in *WHICEB*, 2016, p. 52.
[164] A. Noack, "Modularity clustering is force-directed layout," *Physical Review E*, vol. 79, no. 2, p. 026102, 2009.
[165] D. McGinn, D. Birch, D. Akroyd, M. Molina-Solana, Y. Guo, and W. J. Knottenbelt, "Visualizing dynamic Bitcoin transaction patterns," *Big data*, vol. 4, no. 2, pp. 109-119, 2016.
[166] B. Jenny *et al.*, "Force-directed layout of origin-destination flow maps," *International Journal of Geographical Information Science*, pp. 1-20, 2017.
[167] A. S. Rose, A. R. Bradley, Y. Valasatava, J. M. Duarte, A. Prlić, and P. W. Rose, "Web-based molecular graphics for large complexes," in *Proceedings of the 21st international conference on Web3D technology*, 2016, pp. 185-186: ACM.
[168] P. Kerpedjiev, S. Hammer, and I. L. Hofacker, "Forna (force-directed RNA): Simple and effective online RNA secondary structure diagrams," *Bioinformatics*, vol. 31, no. 20, pp. 3377-3379, 2015.
[169] S. Bang, J. Choi, J. Park, and S.-J. Park, "A hub-protein based visualization of large protein-protein interaction networks," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, 2007, pp. 1217-1220: IEEE.
[170] J. Tuikkala, H. Vähämaa, P. Salmela, O. S. Nevalainen, and T. Aittokallio, "A multilevel layout algorithm for visualizing physical and genetic interaction networks, with emphasis on their modular organization," *BioData mining*, vol. 5, no. 1, p. 2, 2012.
[171] D. C. Fung, M. R. Wilkins, D. Hart, and S. H. Hong, "Using the clustered circular layout as an informative method for visualizing protein–protein interaction networks," *Proteomics*, vol. 10, no. 14, pp. 2723-2727, 2010.
[172] T. Hruz *et al.*, "A multilevel gamma-clustering layout algorithm for visualization of biological networks," *Advances in bioinformatics*, vol. 2013, 2013.
[173] T. Jansen *et al.*, "Mitochondrial DNA and the origins of the domestic horse," *Proceedings of the National Academy of Sciences*, vol. 99, no. 16, pp. 10905-10910, 2002.
[174] A. Teacher and D. Griffiths, "HapStar: automated haplotype network layout and visualization," *Molecular Ecology Resources*, vol. 11, no. 1, pp. 151-153, 2011.
[175] W. Basalaj and K. Eilbeck, "Straight-line drawings of protein interactions," in *Graph Drawing*, 1999, pp. 259-266: Springer.
[176] T. Pisanski, B. Plestenjak, and A. Graovac, "NiceGraph Program and its applications in chemistry," *Croatica Chemica Acta*, vol. 68, no. 1, pp. 283-292, 1995.
[177] R. Hoppe and J. Köhler, "SCHLEGEL projections and SCHLEGEL diagrams-new ways to describe and discuss solid state compounds," *Zeitschrift für Kristallographie-Crystalline Materials*, vol. 183, no. 1-4, pp. 77-112, 1988.
[178] C. Friedrich and F. Schreiber, "Visualisation and navigation methods for typed protein-protein interaction networks," *Applied bioinformatics*, vol. 2, pp. S19-S24, 2003.
[179] B. Genc and U. Dogrusoz, "A constrained, force-directed layout algorithm for biological pathways," in *International Symposium on Graph Drawing*, 2003, pp. 314-319: Springer.
[180] U. Dogrusoz, E. Giral, A. Cetintas, A. Civril, and E. Demir, "A compound graph layout algorithm for biological pathways," in *International Symposium on Graph Drawing*, 2004, pp. 442-447: Springer.
[181] B. Genç and U. Dogrusoz, "A layout algorithm for signaling pathways," *Information Sciences*, vol. 176, no. 2, pp. 135-149, 2006.
[182] K. Kojima, M. Nagasaki, E. Jeong, M. Kato, and S. Miyano, "An efficient grid layout algorithm for biological networks utilizing various biological attributes," *BMC bioinformatics*, vol. 8, no. 1, p. 76, 2007.
[183] R. Santamaría, R. Therón, and L. Quintales, "BicOverlapper: a tool for bicluster visualization," *Bioinformatics*, vol. 24, no. 9, pp. 1212-1213, 2008.
[184] R. Santamaría, R. Therón, and L. Quintales, "A visual analytics approach for understanding biclustering results from microarray data," *BMC bioinformatics*, vol. 9, no. 1, p. 247, 2008.

[185] T. Nugent and D. T. Jones, "Predicting transmembrane helix packing arrangements using residue contacts and a force-directed algorithm," *PLoS computational biology*, vol. 6, no. 3, p. e1000714, 2010.
[186] J.-J. Tsay, B.-L. Wu, and Y.-S. Jeng, "Hierarchically organized layout for visualization of biochemical pathways," *Artificial intelligence in medicine*, vol. 48, no. 2, pp. 107-117, 2010.
[187] A. Crippa, L. Cerliani, L. Nanetti, and J. B. Roerdink, "Heuristics for connectivity-based brain parcellation of SMA/pre-SMA through force-directed graph layout," *Neuroimage*, vol. 54, no. 3, pp. 2176-2184, 2011.
[188] I. F. Akyildiz, "A survey on sensor networks," presented at the IEEE Communications Magazine, 2002.
[189] S.-H. Cheong, K.-I. Lee, Y.-W. Si, and L. H. U, "Lifeline: Emergency Ad Hoc Network," presented at the Computational Intelligence and Security (CIS), 2011 Seventh International Conference on, Hainan, 2011.
[190] O. O'Brien. (2018, September 6). *Bike Share Map*. Available: http://bikes.oobrien.com
[191] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE network*, vol. 15, no. 6, pp. 30-39, 2001.
[192] X. Wang, S. Wang, and J.-J. Ma, "An improved co-evolutionary particle swarm optimization for wireless sensor networks with dynamic deployment," *Sensors*, vol. 7, no. 3, pp. 354-370, 2007.
[193] J. Ma, Q. Liu, and W. Xie, "An Improved Virtual Force-Directed Particle Swarm Optimization Positioning Algorithm," *International Journal of Control and Automation*, vol. 9, no. 5, pp. 1-10, 2016.
[194] Z. F. Islam, M. Romanuik, S. S. Heydari, and M. Salmanian, "OLSR-based coarse localization in tactical MANET situational awareness systems," in *Communications (ICC), 2012 IEEE International Conference on*, 2012, pp. 6494-6498: IEEE.
[195] A. Efrat, D. Forrester, A. Iyer, S. G. Kobourov, C. Erten, and O. Kilic, "Force-directed approaches to sensor localization," *ACM Transactions on Sensor Networks*, vol. 7, no. 3, p. 27, 2010, Art. no. 27.
[196] S.-H. Cheong and Y.-W. Si, "Accelerating the Kamada-Kawai algorithm for boundary detection in a mobile ad hoc network," *ACM Transactions on Sensor Networks*, vol. 13, no. 1, 2016, Art. no. 3.
[197] W. T. Tutte, "How to draw a graph," *Proceedings of the London Mathematical Society*, vol. 3, no. 1, pp. 743-767, 1963.
[198] W. T. Tutte, "Convex representations of graphs," *Proceedings of the London Mathematical Society*, vol. 3, no. 1, pp. 304-320, 1960.
[199] L. Sundström, "A Force Directed Placement Method Including Angular Resolution and Bond Overlap," ed, 2016.
[200] R. Hooke, "1678, Lectures de potentia restitutiva, or of spring explaining the power of springing bodies," *Printed for John Martyn printer to the Royal Society, Bell in St. Paul's church-yard*, 1931.
[201] E. R. Gansner, Y. Koren, and S. North, "Graph drawing by stress majorization," in *International Symposium on Graph Drawing*, 2004, pp. 239-250: Springer.
[202] Y. Koren and A. Civril, "The Binary Stress Model for Graph Drawing," in *Graph Drawing*, 2008, pp. 193-205: Springer.
[203] T. Dwyer, Y. Koren, and K. Marriott, "Constrained graph layout by stress majorization and gradient projection," *Discrete Mathematics*, vol. 309, no. 7, pp. 1895-1908, 2009.
[204] L. Chen and A. Buja, "Stress functions for nonlinear dimension reduction, proximity analysis, and graph drawing," *Journal of Machine Learning Research*, vol. 14, no. Apr, pp. 1145-1173, 2013.
[205] Y.-J. Ko and H.-C. Yen, "Drawing clustered graphs using stress majorization and force-directed placements," in *Information Visualisation (IV), 2016 20th International Conference*, 2016, pp. 69-74: IEEE.
[206] H. Youssef, S. M. Sait, and H. Adiche, "Evolutionary algorithms, simulated annealing and tabu search: a comparative study," *Engineering Applications of Artificial Intelligence*, vol. 14, no. 2, pp. 167-181, 2001.
[207] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087-1092, 1953.
[208] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671-680, 1983.
[209] A. Dekkers and E. Aarts, "Global optimization and simulated annealing," *Mathematical Programming*, vol. 50, no. 1-3, pp. 367-393, 1991.
[210] M. Kudelka, P. Kromer, M. Radvansky, Z. Horak, and V. Snasel, "Efficient visualization of social networks based on modified Sammon′ s mapping," *Swarm and Evolutionary Computation*, vol. 25, pp. 63-71, 2015.
[211] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on computers*, vol. 100, no. 5, pp. 401-409, 1969.
[212] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to parallel computing: design and analysis of algorithms*. Benjamin/Cummings Redwood City, 1994.
[213] M. J. Quinn, *Parallel computing: theory and practice*. McGraw-Hill, Inc., 1994.

[214] J. Zhong and B. He, "GViewer: GPU-accelerated graph visualization and mining," *Social Informatics*, pp. 304-307, 2011.
[215] A. Godiyal, J. Hoberock, M. Garland, and J. C. Hart, "Rapid multipole graph drawing on the GPU," in *International Symposium on Graph Drawing*, 2008, pp. 90-101: Springer.
[216] D. Zhu, K. Wu, D. Guo, and Y. Chen, "Parallelized force-directed edge bundling on the GPU," in *Distributed Computing and Applications to Business, Engineering & Science (DCABES), 2012 11th International Symposium on*, 2012, pp. 52-56: IEEE.
[217] V. Uher, P. Gajdo, and V. Snáel, "The Visualization of Large Graphs Accelerated by the Parallel Nearest Neighbors Algorithm," in *Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on*, 2016, pp. 9-16: IEEE.
[218] A. Arleo, W. Didimo, G. Liotta, and F. Montecchiani, "Large graph visualizations using a distributed computing platform," *Information Sciences*, vol. 381, pp. 124-141, 2017.
[219] B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems," *NCM*, vol. 9, pp. 44-51, 2009.
[220] G. Krijnen, "Accelerating Fruchterman-Reingold with OpenCL," 2014.
[221] Y.-X. Wang, Z.-Z. Li, L. Yao, W. Cao, and Z.-H. Wang, "Two improved GPU acceleration strategies for force-directed graph layout," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, 2010, vol. 13, pp. V13-132-V13-136: IEEE.
[222] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in science & engineering*, vol. 12, no. 3, pp. 66-73, 2010.
[223] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789-828, 1996.
[224] C. Mueller, D. P. Gregor, and A. Lumsdaine, "Distributed Force-Directed Graph Layout and Visualization," *EGPGV*, vol. 6, pp. 83-90, 2006.
[225] A. Tikhonova and K.-L. Ma, "A scalable parallel force-directed graph layout algorithm," in *Proceedings of the 8th Eurographics conference on Parallel Graphics and Visualization*, 2008, pp. 25-32: Eurographics Association.
[226] R. Hadany and D. Harel, "A multi-scale algorithm for drawing graphs nicely," *Discrete Applied Mathematics*, vol. 113, no. 1, pp. 3-21, 2001.
[227] C. Walshaw, "A multilevel algorithm for force-directed graph drawing," presented at the International Symposium on Graph Drawing, 2000.
[228] S. Hachul and M. Jünger, "Drawing large graphs with a potential-field-based multilevel algorithm," in *International Symposium on Graph Drawing*, 2004, pp. 285-295: Springer.
[229] C. Crawford, C. Walshaw, and A. Soper, "A multilevel force-directed graph drawing algorithm using multilevel global force approximation," in *Information Visualisation (IV), 2012 16th International Conference on*, 2012, pp. 454-459: IEEE.
[230] F. G. Toosi and N. S. Nikolov, "Vertex-neighboring multilevel force-directed graph drawing," in *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, 2016, pp. 002996-003001: IEEE.
[231] W. S. Torgerson, "Multidimensional scaling: I. Theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401-419, 1952.
[232] T. Sano, "Automatically undirected graph drawing method with consideration of graph structure," *Systems and computers in Japan*, vol. 28, no. 12, pp. 33-42, 1997.
[233] D. Harel and Y. Koren, "Graph drawing by high-dimensional embedding," *J. Graph Algorithms Appl.*, vol. 8, no. 2, pp. 195-214, 2004.
[234] T. Dwyer, Y. Koren, and K. Marriott, "Stress majorization with orthogonal ordering constraints," in *International Symposium on Graph Drawing*, 2005, pp. 141-152: Springer.
[235] M. Chalmers, "A linear iteration time layout algorithm for visualising high-dimensional data," in *Proceedings of the 7th conference on Visualization'96*, 1996, pp. 127-ff.: IEEE Computer Society Press.
[236] Y. Koren, L. Carmel, and D. Harel, "ACE: A fast multiscale eigenvectors computation for drawing huge graphs," in *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, 2002, pp. 137-144: IEEE.
[237] S. Lespinats, M. Verleysen, A. Giron, and B. Fertil, "DD-HDS: A method for visualization and exploration of high-dimensional data," *IEEE transactions on Neural Networks*, vol. 18, no. 5, pp. 1265-1279, 2007.
[238] L. Chen and A. Buja, "Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis," *Journal of the American Statistical Association*, vol. 104, no. 485, pp. 209-219, 2009.
[239] P. Gajer, M. Goodrich, and S. Kobourov, "A multi-dimensional approach to force-directed layouts of large graphs," in *Graph Drawing*, 2001, pp. 211-221: Springer.

[240] T. Dwyer, K. Marriott, and M. Wybrow, "Integrating edge routing into force-directed layout," in *International Symposium on Graph Drawing*, 2006, pp. 8-19: Springer.

[241] W. Dzwinel, R. Wcisło, and W. Czech, "ivga: A fast force-directed method for interactive visualization of complex networks," *Journal of Computational Science*, 2016.

APPENDIX

Year	Aesthetic drawings for general networks	Biological network visualisation	Node placement and localisation for sensor networks	Information visualisation	Component placement in VLSI circuits design	Scheduling in VLSI circuits design
1963	1	0	0	0	0	0
1964	0	0	0	0	0	0
1965	0	0	0	0	2	0
1966	0	0	0	0	0	0
1967	0	0	0	0	0	0
1968	0	0	0	0	0	0
1969	0	0	0	0	0	0
1970	0	0	0	0	0	0
1971	0	0	0	0	2	0
1972	0	0	0	0	0	0
1973	0	0	0	0	0	0
1974	0	0	0	0	2	0
1975	0	0	0	0	0	0
1976	0	0	0	0	0	0
1977	0	0	0	0	0	0
1978	0	0	0	0	0	0
1979	0	0	0	0	2	0
1980	0	0	0	0	0	0
1981	0	0	0	0	2	0
1982	0	0	0	0	4	0
1983	0	0	0	0	2	0
1984	0	0	0	0	0	0
1985	0	0	0	0	2	0
1986	0	0	0	0	2	0
1987	0	0	0	0	2	6
1988	0	0	0	0	2	0
1989	2	0	0	0	0	12
1990	0	0	0	0	0	0
1991	1	0	0	0	2	18
1992	0	0	0	0	0	6
1993	0	0	0	0	0	0
1994	2	0	0	0	0	0
1995	7	5	0	0	0	6
1996	3	0	0	0	0	6
1997	2	0	0	0	0	0
1998	1	0	0	0	2	0
1999	3	5	0	0	0	6
2000	4	0	0	0	0	12
2001	8	5	0	0	4	0
2002	3	0	0	0	4	6
2003	1	10	0	0	6	0
2004	5	10	4	0	4	0
2005	7	0	0	3	1	6
2006	4	5	0	3	6	6
2007	2	10	8	0	4	12
2008	4	10	0	15	4	0
2009	3	0	0	12	0	0
2010	5	15	4	0	2	0
2011	3	10	0	9	2	0
2012	6	5	4	9	4	6
2013	6	10	0	18	0	6
2014	5	5	0	12	0	0
2015	8	5	0	15	4	0
2016	6	5	4	24	2	0
2017	1	0	0	3	2	0

Figure 8 paper submission count of force-directed algorithms classified by application fields.