Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 6208509, 24 pages
https://doi.org/10.1155/2018/6208509

Research Article

A Force-Directed Algorithm for Drawing Directed Graphs Symmetrically

Taihua Xu,[1,2] **Jie Yang**,[2,3] **and Guanglei Gou**[1,4]

[1]*School of Information Science and Technology, Southwest Jiaotong University, Chengdu, Sichuan 610031, China*
[2]*Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China*
[3]*School of Physics and Electronic Science, Zunyi Normal University, Zunyi 563002, China*
[4]*School of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China*

Correspondence should be addressed to Jie Yang; 530966074@qq.com

Received 27 March 2018; Revised 8 September 2018; Accepted 28 October 2018; Published 19 November 2018

Academic Editor: Eckhard Hitzer

Symmetry is one of the most important aesthetic criteria on graph drawing. It is quite necessary to measure the extent to which the drawings can be considered symmetric. For this purpose, a symmetric metric based on vertex coordinate calculation is proposed in this paper. It is proven theoretically and experimentally that the proposed metric is robust to contraction, expansion, and rotation of drawings. This robustness conforms to human perception of symmetry. Star-subgraphs and cycles are two common structures in digraphs. Both of them have inherent symmetry which should be displayed in drawings. For this purpose, a force-directed algorithm named FDS is proposed which can draw star-subgraphs and cycles as symmetrically as possible. FDS algorithm draws cycles as circles whose positions are fixed to provide a scaffolding for overall layout, renders non-leaf vertices by a standard force-directed layout, and places leaf vertices on concentric circles via a deterministic strategy. A series of experiments are carried out to test FDS algorithm. The results show that FDS algorithm draws digraphs more symmetrically than the existing state-of-the-art algorithms and performs efficiency comparable to $O(n \log n)$ YFHu algorithm.

1. Introduction

Graph drawing is the research on how to communicate knowledge visually through drawings of graphs. The main purpose of graph drawing is to produce understandable drawings from graphs within bearable time. The understandability of drawings is a highly subjective matter and affected by not only intrinsic data characteristics [1], but also drawings themselves. The understandability affected by intrinsic data characteristics can be interpreted as whether the important structures, e.g., star-subgraphs and cycles, are drawn to be eye-catching in drawings. The understandability affected by drawings is specified as aesthetics. Bhanji et al. [2] summed up three common aesthetics: maximized symmetry, minimized edge crossings, and minimized bends. With the development of research on aesthetics, the number of common aesthetics increases to seven [3]. Furthermore, Ware et al. [4] believed that continuity (i.e., keeping multiedge paths as straight as possible) is also an important aesthetic. It is commonly accepted that a drawing would be understandable if it conforms to these aesthetics.

Indubitably, symmetry is one of the most important aesthetic criteria that represents the structure and properties of a graph visually. The importance of symmetry on graph drawing has been pointed out by Lipton et al. [5]. Even Eades and Hong discussed symmetric graph drawing in a chapter of the book "Handbook of Graph Drawing and Visualization" [6]. Existing studies explored the issues of symmetry by combining with isomorphisms [7] and automorphisms [5, 8–10]. In addition, there also is an interesting research topic on symmetry in layouts sketched by participants [11–14]. However, symmetry is on the basis of human intuition and subjective judgement of researchers. In other words, symmetry would remain subjective if there is no quantified metric to measure it. There have been some objective metrics proposed for symmetry. A model for measuring the symmetry of straight line drawing was given by Lipton et al. [5]. But this model is associated with automorphisms which is computational hardly. Purchase [3] presented a formal metric to measure the extent to which the drawing can be considered symmetric by returning an objective real number between 0 and 1 inclusive. Although Purchase's method is applicable to any drawing of any size, it ignored rotational symmetry. And the ignorance is in contradiction with human perception of symmetry. Thus, it is very necessary to develop an objective symmetric metric which is easy to compute and in accord with human perception of symmetry.

Then the first contribution of this paper is that an objective symmetric metric is proposed to measure the extent to which the drawings produced by graph drawing algorithms can be considered symmetric, which is based on vertex coordinate calculation. Locally, this metric works out a value for every vertex, which can measure the extent to which neighbors can be considered to be distributed symmetrically around the vertex. Globally, expected value and variance of all local values are used to measure the extent to which the drawing can be considered symmetric. Compared to these existing symmetry metrics, the proposed symmetric metric in this paper is not only applicable to any drawing of any size, but also robust to contraction, expansion, and rotation of drawings. In addition, it can return metric values within few time because it is based on vertex coordinate calculation.

As two common structures in digraphs, a star-subgraph contains a star vertex and several leaf vertices connecting the star vertex, and a cycle is defined as a closed path with no repetitions of vertices and edges. From viewpoint of knowledge visualization, star-subgraphs and cycles are worthy of being visualized since the two structures from digraphs are originally two kinds of knowledge. It should be noted that there is inherent symmetry in star-subgraphs and cycles, so these two structures should be displayed symmetrically in overall drawings. For drawing cycles, Becker and Rojas [15] once tried to draw cycles by circular layout algorithm. Although circular layout algorithm can display cycles as circles which is the most symmetrical geometric figure, it can not highlight circles. As a result, the circles can not be easily distinguished from overall drawings. On the other hand, the task of drawing star-subgraphs is essentially how to distribute leaf vertices around star vertices. There are already some works which involve leaf vertices. In multilevel force-directed approaches [16], they deal with leaf vertices by deletion. However, it is inappropriate from the viewpoint of knowledge discovery because removing of leaf vertices would cause loss of knowledge. The problem of drawing star-subgraphs symmetrically is actually the problem of drawing leaf vertices. Some multilevel force-directed approaches can display leaf vertices via coarsening phase [17]. However, displaying of these leaf vertices still takes noticeable time and does not exhibit sufficient inherent symmetry of star-subgraphs. RINGs [18], Circular [19], and Radial [20] layouts can also deal with leaf vertices. RINGs algorithm [18] places successors in concentric rings around the center of the predecessor circle for any digraphs. However, overlapping between vertices would be induced when other structures exist except for star-subgraphs. In the drawing of star-subgraphs, produced by Circular algorithm [19], vertices are placed on the same circle leading to insufficient space utilization. Star-subgraphs can be drawn with the shape of fan by Radial algorithm [20], which did not exhibit sufficient inherent symmetry of star-subgraphs. Thus, there is an urgent need for an algorithm which can draw digraphs containing star-subgraphs and cycles symmetrically conspicuously.

In consideration of the demand for displaying star-subgraphs and cycles symmetrically, and based on above discussions, the second contribution of this paper is that a force-directed algorithm named FDS is proposed by using scaffolding strategy, which can solve the problems encountered in drawing star-subgraphs and cycles. The reason for taking scaffolding strategy is that the positions of cycle vertices should be fixed once the coordinates of them are acquired so that the symmetry displayed are not changed in the process of drawing other vertices. Similar scaffolding strategy has been used in [21, 22]. In [21], authors took spanning trees and planar graphs to provide a scaffolding for drawing so that structure and follow path can be discerned. Unlike [21], in this paper, we take the found cycles to provide the scaffolding for overall drawing so that cycles can be discerned from overall drawing. The basic idea of FDS is as follows: firstly, use HL-DPC algorithm [23] to discover cycles of digraphs, and a cycle drawing algorithm based on circle placement (CD-CP) is proposed as a subalgorithm of FDS. CD-CP algorithm can fix the cycle vertices on equal amount of highlighted circles which provide a fixed scaffolding to overall drawings. Secondly, a standard force-directed algorithm, FR algorithm [24], is taken to deal with non-leaf vertices. Thirdly, a leaf vertices distribution algorithm (LD) is proposed as a subalgorithm of FDS. LD algorithm can distribute leaf vertices evenly around the centers of star-subgraphs and star vertices, so that the drawing of star-subgraphs is symmetrical. Compared to existing methods, our FDS algorithm solves the problems when drawing cycles and star-subgraphs. For cycles, as FDS's subalgorithm, CD-CP algorithm can not only distribute cycle as circles but also highlight them so that cycles are displayed symmetrically and conspicuously. For star-subgraphs, as FDS's subalgorithm, LD algorithm can distribute leaf vertices belonging to star-subgraphs most symmetrically by cost of few time.

It can be seen that a standard force-directed algorithm, FR, is used as a subalgorithm of proposed FDS algorithm. The reason for utilizing FR algorithm is that force-directed algorithms are the most widely used tools for graph drawing and famous for ability of producing pleasing layouts having balanced aesthetics. The basic idea of force-directed algorithm is that a graph is simulated by a physical system of attractive spring forces along edges and repulsive forces emanating from vertices. Then vertices will move along the direction of the combined force. These movements are repeated until the force system reaches equilibrium. There are already various force-directed algorithms. In 1984, Eades [26] proposed a landmark force-directed algorithm. However, Eades's implementation does not follow Hook law but the spring force formula built by himself. Kamada and Kawai (KK) [27] proposed an energy model which made improvement on Eades's spring model. KK algorithm uses spring forces proportional to the graph theoretic distances. Appealing drawing can be obtained by decreasing the system total energy until minimum. Fruchterman and Reingold (FR) [24] referenced the works of Eades [26] and Quinn [28]. They used a spring-electrical system, which is based on Coulomb's law, to take the place of Eades's spring physical system. Although FR is a very simple algorithm, it provides quite excellent results. Even today, it is still one of the most popular algorithms for graph drawing. Jacomy [29] proposed the well-known forceatlas2 algorithm by reference of LinLog model. Forceatlas2 algorithm can produce good quality with few iterations for most graphs [30]. An important improvement on force-directed algorithms is the multilevel technique. There are some famous multilevel graph algorithms, such as FMS [31], GRIP [32], Walshaw's [33], FM^3 [16], and YifanHu [17] algorithms. YifanHu algorithm is both efficient and high quality, which is proposed based on combining a multilevel approach with the Barnes and Hut [34] octree technique. There are other outstanding algorithms, such as Gansner's stress majorization [35], Landmark MDS [36], PivotMDS [37], and MaxEnt [38].

This paper is organized as follows. In Section 2, related definitions and works are given. In Section 3, a symmetric metric based on vertex coordinate calculation is proposed to measure the extent to which the drawings produced by various algorithms can be considered symmetric. In Section 4, the proposed FDS algorithm is described in detail. In Section 5, a Gephi layout plugin is developed in NetBeans that is a software development platform written in Java. Related experiments are performed to compare the efficiency and quality of FDS algorithm with existing methods, including FR algorithm [24], YifanHu algorithm [17], and ForceAtlas2 algorithm [29]. Finally, in Section 6 a conclusion is given.

2. Related Work

In this section, firstly, we briefly review several related graph definitions; secondly, HL-DPC algorithms [23] should be reviewed shortly because it is used to compute cycles of digraphs; thirdly, FR algorithm [24], a standard force-directed algorithm, also should be introduced briefly because we take it to draw non-leaf vertices of digraphs.

2.1. Related Graph Definitions. In order to facilitate the descriptions in this paper, related definitions of graph theory are introduced here. A graph may be either directed or undirected. A digraph is a pair $D = \{V, E\}$, where V represents the set of vertices and E the set of directed edges. A *leaf vertex* is a vertex with degree one. A *universal vertex* is a vertex that is adjacent to every other vertex in the graph. A *star vertex* is a vertex that is adjacent to several *leaf vertices*. A *star vertex* and its adjacent leaf vertices can derive a *star-subgraph*. If a closed path is with no repetitions of vertices and edges, then the path is called a directed *cycle*. As the two notions, *leaf vertices* and subgraphs, are involved in this work, related definitions are also introduced as follows.

Definition 1 (see [39]). Let $D = \{V, E\}$ be a simple graph. The subgraph induced by a subset W of the vertex set V is the graph $\{W, F\}$, where the edge set F contains an edge in E if and only if both endpoints of this edge are in W.

Definition 2. Let $D = \{V, E\}$ be a digraph, with V the set of vertices and E the set of edges, where $V = \{v_1, v_2, \ldots, v_{n-1}, v_n\}$. The number of *leaf vertices* associated with v_i is denoted by $leaf\_number(i)$.

Definition 3. Let $D = \{V, E\}$ be a digraph. The number of *leaf vertices* in D is denoted by $n_{leaf}(D)$. Then the proportion of *leaf vertices* in D is denoted by $p_{leaf}(D)$, which is calculated by

$$p_{leaf}(D) = \frac{n_{leaf}(D)}{|V|}. \quad (1)$$

2.2. HL-DPC Algorithms [23]. One of the purposes of this paper is to draw cycles as symmetric as possible. The precondition of drawing cycles is finding cycles. In [23], HL-DPC algorithm is proposed to find cycles from strongly connected components (SCCs) with 3 vertices at least which is proven to contain cycles certainly. The main idea of the HL-DPC algorithm is to conduct heuristic DFS to a given SCC which is found from a digraph via Tarjan's algorithm [40]. There are two pieces of heuristic information to decide whether DFS is forward or backward and control the process of DFS on choosing vertices. A stack S is used for recording the vertices visited by heuristic DFS. If a vertex u belonging to S is visited again but the size of S does not meet the experimental threshold, then u would be removed from S. DFS continues to run from the top vertex in S until visiting vertex is visited again and the size of S meets an experimental threshold. Then a cycle is asserted to be found.

The two pieces of heuristic information used in HL-DPC algorithm are introduced as follows. Firstly, a direction parameter is introduced to decide whether DFS is forward or backward.

Definition 4 (see [23]). Let $D = \{V, E\}$ be a digraph. $D^3_{scc} = \{V^3_{scc}, E^3_{scc}\}$ is a SCC of D with 3 vertices at least. A direction parameter, *ratio*, is defined by

$$ratio = \frac{\prod_{v \in V^3_{scc}} S\deg^+(v)}{\prod_{v \in V^3_{scc}} S\deg^-(v)}. \quad (2)$$

In (2), $S\deg^+(v)$ and $S\deg^-(v)$ are the number of outgoing edges and incoming edges adjacent to vertex v in the SCC, respectively. Then according to the value of *ratio*, the direction of DFS can be determined by the corresponding formula in [23]. The experiments in [23] also show that the proposed parameter is necessary and useful.

Secondly, DFS will visit neighbor vertex with maximum $S\deg^+(v)$ and $S\deg^-(v)$ of visited vertex v. The second heuristic information is also proved to be necessary by corresponding experiments in [23].

2.3. FR Algorithm [24]. In this paper, we take a standard force-directed algorithm as a part of our methods. It is necessary to review it briefly. Force-directed algorithms model the graph drawing problem by calculating attractive and repulsive forces between vertices. Optimal drawing would be achieved when the energy of system descends to the minimum. Force-directed algorithms can produce appealing drawings for most graphs and display isomorphic and symmetric substructures. Although there is a drawback of high runtime compared to other graph drawing algorithms, force-directed algorithms still dominate the algorithms of graph drawing. FR algorithm which requires $O(n^2)$ time is one of the most famous force-directed algorithms. It was proposed by Fruchterman and Reingold [24] based on the famous Eades's spring force-directed [26] model. Although FR is a very simple algorithm, it provides quite excellent results. Even today, after years of its creation, it is still one of the most popular algorithms for graph drawing. In this work, FR algorithm will be used to calculate repulsive and attractive forces between vertices.

Fruchterman and Reingold represented the graph drawing problem by a system of electrically charged vertices connected by some springs. There are only two criteria demanded for a good graph drawing. (1) Connected vertices should be close to each other. (2) Vertices should not be drawn too close to each other.

Certainly FR model also moves vertices according to the attractive and repulsive forces. When the total energy of the system decreases to a minimum, the movement will stop and the best drawing will be achieved. The repulsive force, f_r, exists between any different two vertices v_i and v_j. f_r is inversely proportional to the distance between v_i and v_j. The attractive force, f_a, just exists between linked vertices. f_a is proportional to the square of the distance:

$$\begin{aligned} f_r &= \frac{k^2}{d} u(2k-d), \\ f_a &= \frac{d^2}{k}, \end{aligned} \tag{3}$$

where $u(x) = \{1, \text{if } x > 0; 0, \text{otherwise}\}$.

In (3), k is optimal length [24], or natural spring length [33]. It is calculated by $k = C\sqrt{(area/|V|)}$. The constant C is found experimentally and *area* restricts the field of moving vertices. And d is the distance between vertices v_i and v_j.

Then combined force on a vertex v_i and the total energy of the system are given in (4) and (5), respectively.

$$\begin{aligned} f(i,k) = &\sum_{i \neq j} \frac{-k^2}{\left\| c_j - c_i \right\|^2} \left(c_j - c_i \right) \\ &+ \sum_{i \longleftrightarrow j} \frac{\left\| c_j - c_i \right\|}{k} \left(c_j - c_i \right), \end{aligned} \tag{4}$$

where $i \longleftrightarrow j$ means vertices v_i and v_j are neighboring vertices.

$$Energy_{sys}(k) = \sum_{i \in V} f^2(i,k). \tag{5}$$

In [17], the author has proved theoretically that changing of parameters k does not actually change the minimal energy drawing of the graph but merely scales the drawing from a mathematical point of view.

3. A Symmetric Metric for Graph Drawing

In this section, a new objective symmetric metric based on vertex coordinate calculation is proposed to measure the extent to which the drawings produced by graph drawing algorithms can be considered symmetrical.

Definition 5. Let $D = \{V, E\}$ be a digraph, with V the set of vertices and E the set of edges, where $V = \{v_1, v_2, \ldots, v_{n-1}, v_n\}$. In the drawing of D, the coordinate of vertex $v_i (i = 1, 2, \ldots, n)$ is denoted as (v_{ix}, v_{iy}). Vertex v_i and its neighbors $N_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ can be grouped into a subset $W_i = \{v_i, v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ of V. The barycenter of W_i is denoted as $G(W_i)$. The center of minimum circumscribed circle, min-circumcenter, of W_i is denoted as $O^{MCC}(W_i)$. The radius of minimum circumscribed circle, min-circumradius, of W_i is denoted as $r^{MCC}(W_i)$. Locally, the metric $\sigma(i)$ in (6) is used to measure the extent to which the drawing of W_i related to vertex v_i can be considered symmetrical. Note that smaller $\sigma(i)$ is better. Globally, the metrics $E\sigma$ and $D\sigma$ in (7) are used to measure the extent to which overall drawing can be considered symmetrical. Note that $E\sigma$ is the primary metric in contrast to $D\sigma$. For $E\sigma$ and $D\sigma$, the smaller values are better.

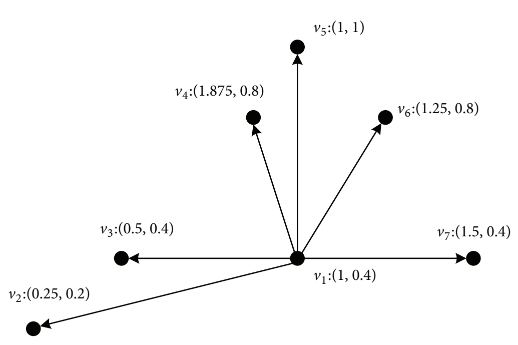
$$\sigma(i) = \frac{dis\left| G(W_i), O^{MCC}(W_i) \right|}{r^{MCC}(W_i)}. \tag{6}$$

$$\begin{aligned} E\sigma &= \frac{1}{n} \sum_{i=1}^{n} \sigma(i), \\ D\sigma &= \frac{1}{n} \sum_{i=1}^{n} \left[E\sigma - \sigma(i) \right]^2. \end{aligned} \tag{7}$$

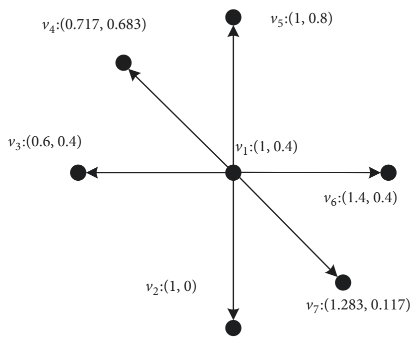
In (6), $dis|a,b|$ is the Euclidean distance between a and b which is easy to be obtained. Firstly, it is needed to explain how the minimum circumscribed circle of a set of several vertices can be determined. The convex hull of the drawing of W_i is found by Graham's scan [25]. Then three vertices on convex hull are chosen via exhaustive search to determine the minimum circumscribed circle of the drawing of W_i. For the three chosen vertices A, B, C, the coordinate of min-circumcenter $(O^{MCC}(W_i)_x, O^{MCC}(W_i)_y)$ and the size of min-circumradius $r^{MCC}(W_i)$ of W_i can be determined by

$$\Delta = 2\left(A_x - B_x\right)\left(C_y - B_y\right) - 2\left(A_y - B_y\right)\left(C_x - B_x\right),$$

$$O^{MCC}(W_i)_x = \frac{\left(C_y - B_y\right)\left(A_x^2 + A_y^2 - B_x^2 - B_y^2\right) - \left(A_y - B_y\right)\left(C_x^2 + C_y^2 - B_x^2 - B_y^2\right)}{\Delta},$$

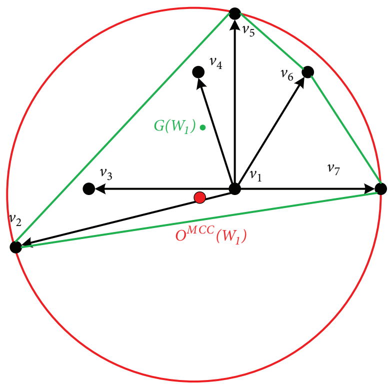


(a) Drawing 1

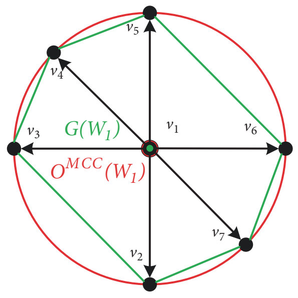


(b) Drawing 2

FIGURE 1: Two different drawings of digraph D.



(a) Drawing 1



(b) Drawing 2

FIGURE 2: Determining the minimum circumscribed circle of the vertex set of W_1 corresponding to vertex v_1 in two drawings of D. Firstly, the convex hull (green lines) of W_1 is found by using of Graham's scan [25]. Secondly, three vertices (v_2, v_5, v_7 in drawing 1 and v_2, v_6, v_7 in drawing 2) on the convex hull are chosen via exhaustive search to determine the minimum circumscribed circle (red circle) of W_1. It is noted that determining a circle is essentially of determining its center (red dot) and radius. In addition, the barycenter of green dot is also marked there.

$$O^{MCC}(W_i)_y = \frac{(A_x - B_x)(C_x^2 + C_y^2 - B_x^2 - B_y^2) - (C_x - B_x)(A_x^2 + A_y^2 - B_x^2 - B_y^2)}{\Delta},$$
$$r^{MCC}(W_i) = dis\left|A, O^{MCC}(W_i)\right|. \quad (8)$$

Example 6 is designed to explain how (6) and (7) measure the symmetry of every vertex and overall drawing.

Example 6. Given a digraph $D = \{V, E\}$, with $V = \{v_1, v_2, \dots, v_7\}$ and $E = \{(v_1, v_2), (v_1, v_3), \dots, (v_1, v_7)\}$, two different drawings of D are shown in Figure 1. And coordinates of vertices in the drawings are also shown.

For vertex v_1, its neighbors $N_1 = \{v_2, v_3, \dots, v_7\}$ and v_1 itself can be grouped into a subset $W_1 = \{v_1, v_2, \dots, v_7\}$ of V. In drawing 1 (Figure 1(a)) of D, one can use $\sigma(1)$ computed by (6) to measure the symmetry of vertex v_1. In fact, the computing of $\sigma(1)$ is based on the computing of barycenter and the determination of the minimum circumscribed circles. The two operations for drawings 1 and 2 are also shown in Figure 2.

The barycenter of W_i, $(G(W_1)_x, G(W_1)_y)$, is calculated by

$$G(W_1)_x = \frac{(1 + 0.25 + 0.5 + 0.875 + 1 + 1.25 + 1.5)}{7} = 0.911,$$
$$G(W_1)_y = \frac{(0.4 + 0.2 + 0.4 + 0.8 + 1 + 0.8 + 0.4)}{7} = 0.571. \quad (9)$$

By exhaustive search, v_2, v_5, v_7 are chosen to determine the minimum circumscribed circles of W_i. Then according to

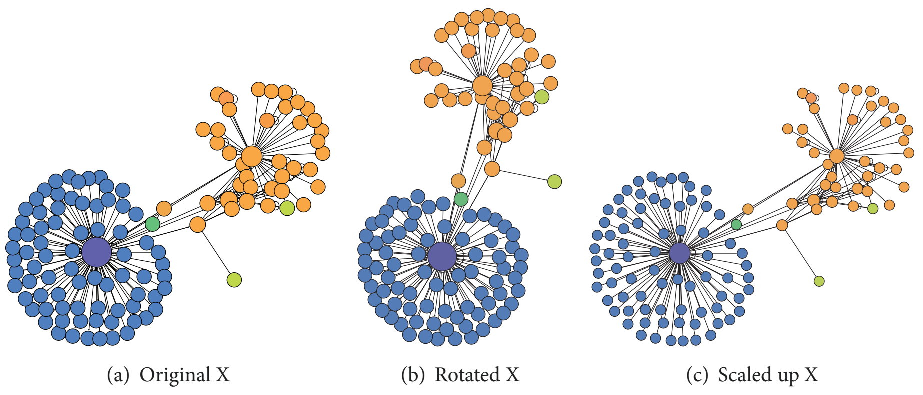


Figure 3: An example drawing named X (a) and its rotated version (b) and its scaled up version (c).

(8), the coordinate of $O^{MCC}(W_i)$ and the size of $r^{MCC}(W_i)$ are calculated by

$$\begin{gathered}\Delta = 1.7,\\ O^{MCC}(W_i)_x = 0.863,\\ O^{MCC}(W_i)_y = 0.377,\\ r^{MCC}(W_i) = 0.638.\end{gathered} \tag{10}$$

By (6), one can get $\sigma(1) = 0.313$. $\sigma(i)$ $(i = 2, 2, \ldots, n)$ can also be calculated in a similar way. As vertex $v_i (i = 2, 2, \ldots, n)$ has only one neighbor, the minimum circumscribed circle of W_i is the circle with diameter of the line segment between v_i and its only neighbor. Thus for W_i, the coordinate of its min-circumcenter is identical to the coordinate of its barycenter. As a result, $\sigma(2) = \sigma(3) = \sigma(4) = \sigma(5) = \sigma(6) = \sigma(7) = 0$. Finally, one can get that $E\sigma = 4.47E-2$ and $D\sigma = 1.2E-2$ by (7).

In drawing 2 (Figure 1(b)) of D, $\sigma(1) = 0$ because the coordinate of min-circumcenter is identical to the coordinate of barycenter for W_i. Similarly, $\sigma(2) = \sigma(3) = \sigma(4) = \sigma(5) = \sigma(6) = \sigma(7) = 0$. Finally, one can get $E\sigma = 0$ and $D\sigma = 0$ for drawing 2.

The comparison of $E\sigma$ indicates that drawing 2 (Figure 1(b)) can be considered more symmetric than drawing 1 (Figure 1(a)). This result is in accord with human intuition.

According to Example 6, the proposed metric seems to be effective for measuring the symmetry displayed in different drawings. For a drawing, if a metric is used to measure how symmetric the drawing is, then how do the metric values change with drawing's rotation and scaling up or down? An example of drawing named X is shown in Figure 3(a). Then the rotated and scaled up X are shown in Figures 3(b) and 3(c), respectively. From the viewpoint of human intuitive impression on the three drawings in Figure 3, the values of symmetric metric on them should be identical. By computing, the values of $E\sigma$ and $D\sigma$ for three drawings in Figure 3 are listed in Table 1. It can be seen that the values of $E\sigma$ and $D\sigma$ for three drawings are equal. Furthermore, Proposition 7 proves theoretically that the proposed symmetric metric is robust for rotation and scaled change of drawings.

Table 1: Symmetric measurement on the three drawings in Figure 3 by the proposed metric.

Drawings	X	Rotated X	Scaled up X
$E\sigma$	1.45E-1	1.45E-1	1.45E-1
$D\sigma$	2.67E-1	2.67E-1	2.67E-1

Proposition 7. *Let $D = \{V, E\}$ be a digraph and X be a drawing of D. For an arbitrary vertex v_i in D, $\sigma(i)$ (6) is used to measure the extent to which neighbors of v_i can be considered to be distributed symmetrically around v_i. And $E\sigma$ and $D\sigma$ (7) are used to measure the extent to which X can be considered symmetrically. Then,*

(1) *$E\sigma$ and $D\sigma$ for X would not change if X is scaled up or down.*

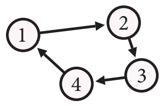
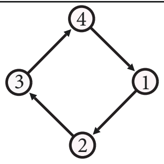
(2) *$E\sigma$ and $D\sigma$ for X would not change if X is rotated at any angle.*

Proof. For arbitrary vertex v_i, its coordinate in X is (v_{ix}, v_{iy}). v_i and its neighbors $N_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ can be grouped into a subset $W_i = \{v_i, v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ of V. Then $\sigma(i) = (dis|G(W_i), O^{MCC}(W_i)|)/r^{MCC}(W_i)$.

(1) After X is scaled up $(k > 1)$ or down $(k < 1)$ k times, the coordinate of v_i in the scaled X changes to be $(k * v_{ix}, k * v_{iy})$. By algebraic simplification, the coordinates of all vertices and auxiliary points (barycenter and min-circumcenter) change in the same way. Let us calculate the value of $\sigma'(i)$ for v_i in the scaled X according to (6) and (8). It is straightforward that $dis|G'(W_i), O^{MCC'}(W_i)| = k * dis|G(W_i), O^{MCC}(W_i)|$ and $r^{MCC'}(W_i) = k * r^{MCC}(W_i)$. Thus,

$$\sigma'(i) = \frac{dis\left|G'(W_i), O^{MCC'}(W_i)\right|}{r^{MCC'}(W_i)}$$

Table 2: Comparisons of $E\sigma$ and $D\sigma$ of one drawing with that of another for a cycle.

	A random drawing of a cycle	Circle drawing of the cycle
Cycle drawings		
$E\sigma$	4.15E-1	3.33E-1
$D\sigma$	5.92E-2	0

$$= \frac{k * dis\left|G(W_i), O^{MCC}(W_i)\right|}{k * r^{MCC}(W_i)}$$
$$= \frac{dis\left|G(W_i), O^{MCC}(W_i)\right|}{r^{MCC}(W_i)} = \sigma(i). \tag{11}$$

This indicates that $\sigma(i)$ is robust for scaled drawings. Finally, according to (7), robustness of $\sigma(i)$ decides the robustness of $E\sigma$ and $D\sigma$. In a word, $E\sigma$ and $D\sigma$ for X would not change if X is scaled up or down.

(2) After X is rotated at angle θ, the coordinate of v_i in the rotated X changes to be $(v_{ix} * \cos\theta - v_{iy} * \sin\theta)$, $v_{ix} * \sin\theta + v_{iy} * \cos\theta)$. By algebraic simplification, the coordinates of all vertices and auxiliary points (barycenter and min-circumcenter) change in the same way. Let us calculate the value of $\sigma''(i)$ for v_i in the rotated X according to (6) and (8). By algebraic simplification again, it can be got that $dis|G''(W_i), O^{MCC''}(W_i)| = dis|G(W_i), O^{MCC}(W_i)|$ and $r^{MCC''}(W_i) = r^{MCC}(W_i)$. Thus,

$$\sigma''(i) = \frac{dis\left|G''(W_i), O^{MCC''}(W_i)\right|}{r^{MCC''}(W_i)}$$
$$= \frac{dis\left|G(W_i), O^{MCC}(W_i)\right|}{r^{MCC}(W_i)} = \sigma(i). \tag{12}$$

This means that $\sigma(i)$ is robust for drawing's rotation at any angle. Globally, robustness of $\sigma(i)$ decides the robustness of $E\sigma$ and $D\sigma$. In a word, $E\sigma$ and $D\sigma$ for X would not change if X is rotated at any angle.

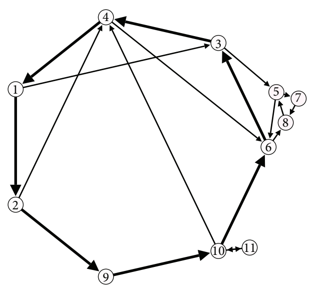
□

4. Proposed Approach

In this section, a force-directed algorithm called FDS is proposed to draw digraphs containing cycles and subgraphs symmetrically and conspicuously. The key term of FDS algorithm is "circle" which is the most symmetrical geometric figure and can be eye-catching from overall drawings. Thus FDS algorithm tries to draw cycle vertices and leaf vertices belonging to star-subgraphs as various kinds of circles which can display most of symmetry from the two structures. The basic idea of FDS is as follows: firstly, use HL-DPC algorithm [23] to discover cycles of digraphs, and a cycle drawing algorithm based on circle placement (CD-CP) is proposed as a subalgorithm of FDS. CD-CP algorithm can fix the cycle vertices on equal amount of highlighted circles which provide a fixed scaffolding to overall drawings. Secondly, a standard force-directed algorithm, FR algorithm [24], is taken to deal with non-leaf vertices. Thirdly, a leaf vertices distribution algorithm (LD) is proposed as a subalgorithm of FDS. LD algorithm can distribute leaf vertices evenly around the centers of star-subgraphs, star vertices, so that the drawings of star-subgraphs are symmetrical. The advantages of FDS algorithm on drawing cycles and star-subgraphs symmetrically are mainly from three aspects: (1) CD-CP algorithm can not only distribute cycle as circles but also highlight them so that cycles are displayed symmetrically and conspicuously; (2) the scaffolding strategy is utilized in FDS algorithm for maintaining the symmetry of cycles displayed in overall drawings; (3) LD algorithm can distribute leaf vertices belonging star-subgraphs most symmetrically by cost of few time.

4.1. A Cycle Drawing Algorithm Based on Circle Placement (CD-CP). In Table 2, two different drawings of a cycle are listed. By comparing $E\sigma$ and $D\sigma$ of one drawing with that of another drawing, it is found that the drawing in which four cycle vertices are arranged on a circle displays more symmetry of the cycle and is more appealing than another. Therefore, in this paper, all cycle vertices are fixed on the same amount as circles with cycle edges being highlighted.

Cycles must be detected before displaying. Thus HL-DPC algorithm [23] is used to detect cycles in digraphs. In the next, it should be elaborated how to place cycle vertices on circles and determined what sizes and location of circles are. (1) If there is only one cycle c, vertices belonging to c are placed evenly on a circle centered at coordinate (0,0), e.g., the cycle in Figure 4(a). (2) If there are x $(x > 1)$ cycles, x points are selected evenly on the circle centered at coordinate (0,0), then cycle vertices are placed evenly on the circle centered at points selected, e.g., the three cycles in Figure 4(b). It is noteworthy that all of cycle edges are set to be 3 times thick of original size so that cycle can be distinguished easily from overall drawing. According to the above description, CD-CP algorithm is proposed as Algorithm 1.


(a) To draw a digraph containing 1 cycle by placing cycle vertices on a circle

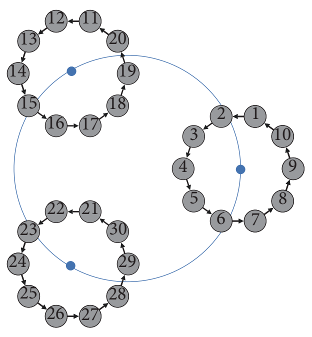

(b) To draw a digraph containing 3 cycles by placing cycle vertices on 3 circles. Blue circle and dots are aides for understanding the placement

Figure 4: Placement of cycle vertices on circles.

```
Input: Cycle sets $C$ // elements of $C$ are cycles and every cycle consists of several cycle vertices
Output: set of coordinates of cycle vertices
1:  Set $R_c$ and $R$ empirically;// $R$ is the radius of circle.
2:  If size($C$)=1 then
3:      $c \longleftarrow$ only cycle in $C$;  $i \longleftarrow 0$;
4:      for each $v$ in $c$ do
5:          $v_x = R_c * \cos(2\pi i/size(c))$; $v_y = R_c * \sin(2\pi i/size(c))$;
6:          $i \longleftarrow i + 1$;
7:      end for
8:      Size of every cycle edge corresponding to $c$ is set to be 3 times thick of original size;
9:  end if
10: if size($C$) > 1 then
11:     $i \longleftarrow 0$, $j \longleftarrow 0$;
12:     for each $c$ in $C$ do // $c$ is a cycle, denote the center of circle for $c$ by $O(c)$
13:         $O(c)_x = R * \cos(2\pi i/size(C))$; $O(c)_y = R * \sin(2\pi i/size(C))$;
14:         $i \longleftarrow i + 1$;
15:         for each $v$ in $c$ do
16:             $v_x = O(c)_x + R_c * \cos(2\pi j/size(c))$; $v_y = O(c)_y + R_c * \sin(2\pi j/size(c))$;
17:             $j \longleftarrow j + 1$;
18:         end for
19:         Size of every cycle edge corresponding to $c$ is set to be 3 times thick of original size;
20:     end for
21: end if
```

Algorithm 1: A cycle drawing algorithm based on circle placement (CD-CP algorithm).

4.2. Leaf Vertices Distribution Algorithm (LD). In order to display star-subgraphs symmetrically, a leaf vertices distribution algorithm (LD) is designed to provide concentric circles on which *leaf vertices* belonging to star-subgraphs can be placed. LD algorithm is proposed based on three calculation steps.

Given a *star vertex* v_i connecting several *leaf vertices*, then,

(1) count the number of *leaf vertices* for v_i;

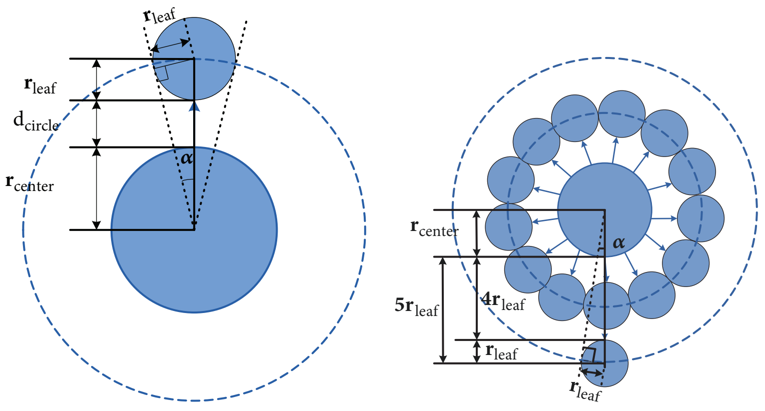
(2) calculate the number of *leaf vertices* that can be placed on every layer of concentric circle around v_i;

(3) calculate the number of concentric circles needed by v_i.

The first step is to count the number of *leaf vertices* for v_i: that is, *leaf_number*(i).

The second step is to calculate the number of *leaf vertices* that can be placed on every layer of concentric circle around v_i.

At first, three notations are introduced. In a drawing, every vertex is drawn to be a solid circle. The radius of the solid circle of v_i is denoted as $r_{center}(i)$. The radius of every *leaf vertex* is r_{leaf}. The distance between borders of *leaf vertex* and v_i is d_{circle}. Generally, the values of r_{leaf} and $r_{center}(i)$ are known already. And d_{circle} can be set artificially to be equal to r_{leaf}. Consequently, the radius of 1st layer circle is $2r_{leaf} + r_{center}(i)$.



$\mathbf{r}_{\text{leaf}}$
$\mathbf{r}_{\text{leaf}}$
$\mathrm{d}_{\text{circle}}$
α
$\mathbf{r}_{\text{center}}$
$\mathbf{r}_{\text{center}}$
α
$5\mathbf{r}_{\text{leaf}}$
$4\mathbf{r}_{\text{leaf}}$
$\mathbf{r}_{\text{leaf}}$
$\mathbf{r}_{\text{leaf}}$

(a) The number of *leaf vertices* can be placed on 1st layer circle around v_i

(b) The number of *leaf vertices* can be placed on 2nd layer circle around v_i

FIGURE 5: The number of *leaf vertices* that can be placed on the 1st and 2nd layer of concentric circles around v_i.

In Figure 5(a), it can be seen that one *leaf vertex* occupies angle of 2α in v_i. Then the number of *leaf vertices* that can be placed on the 1st layer circle around v_i, $a_1(i)$, can be calculated by (13a). Analogously, the radius of the 2nd layer circle is $r_{center}(i)+5r_{leaf}$ as Figure 5(b). And the number of *leaf vertices* that can be placed on 2nd layer circle around v_i is $a_2(i) = \lfloor \pi/\arcsin(r_{leaf}/(5r_{leaf}+r_{center}(i)))\rfloor$. Thus, it can be induced that $a_l(i)$ (l is a positive integer) *leaf vertices* can be placed on the lth circle. And $a_l(i)$ can be calculated by (13b). Generally, $r_{center}(i)$ is an integer multiple of r_{leaf} in practical drawings. Therefore, a more simplified (13c) is proposed to replace (13b).

$$a_1(i) = \left\lfloor \frac{2\pi}{2\alpha} \right\rfloor = \left\lfloor \frac{\pi}{\arcsin\left(r_{leaf}/\left(2r_{leaf}+r_{center}(i)\right)\right)} \right\rfloor, \tag{13a}$$

$$a_l(i) = \left\lfloor \frac{\pi}{\arcsin\left(r_{leaf}/\left((3l-1)\,r_{leaf}+r_{center}(i)\right)\right)} \right\rfloor, \tag{13b}$$

$$a_l(i) = a_1(i) + 9(l-1), \tag{13c}$$

where $\lfloor\bullet\rfloor$ is the largest integer not greater than $\bullet$.

The third step is to calculate the number of concentric circles needed by v_i.

The number of concentric circles needed by v_i is actually the maximum value of l, denoted as L. Considering the condition of known $a_1(i)$ and *leaf_number(i)*, and the fact that the number of *leaf vertices* on every concentric circle can form an arithmetic sequence, L can be calculated by (14) based on the fact that *leaf_number*(i) is less than or equal to the total number of *leaf vertices* that can be placed on concentric circles.

$$L = \left\lceil \frac{\left[9-2a_1(i)+\sqrt{\left[2a_1(i)-9\right]^2+72\,leaf\_number(i)}\right]}{18} \right\rceil, \tag{14}$$

where $\lceil\bullet\rceil$ is the least integer greater than $\bullet$.

It is noteworthy that the number of *leaf vertices* needed to be placed on the last layer circle may not be equal to the number *leaf vertices* that can be placed on the last layer circle. After placement of *leaf vertices* on the $(L-1)$th layer circle, the number of *leaf vertices* not placed yet can be calculated by

$$\begin{aligned} a_{rest\_at\_L} &= leaf\_number(i) - \sum_{j=1}^{L-1} a_j(i) \\ &= leaf\_number(i) - (L-1)\,a_1(i) \\ &\quad - 9(L-1)(L-2). \end{aligned} \tag{15}$$

These three calculation steps bring up LD algorithm as Algorithm 2.

4.3. Proposed FDS Algorithm.

The proposed CD-CP and LD algorithms can draw cycles and star-subgraphs symmetrically locally. However, there are probably other vertices which are neither cycle vertices nor leaf vertices in practical datasets. For these vertices, FR [24] algorithm is taken as its good performance on aesthetics balance. Finally, HL-DPC [23], CD-CP, FR, and LD algorithms are integrated into a whole algorithm called FDS which can draw digraphs symmetrically globally.

Related definitions and propositions needed for introducing FDS algorithm are given below.

Definition 8. Let $D = \{V, E\}$ be a digraph, with V the set of vertices and E the set of edges. $D_f = \{V_f, E_f\}$ is the force-subgraph of D. The mathematical descriptions of E_f and V_f are as (16a) and (16b), respectively. Let $p_{v-force}(D)$ be the proportion of vertices belonging to D_f in D and $p_{e-force}(D)$ be the proportion of edges belonging to D_f in D. $p_{v-force}(D)$ and $p_{e-force}(D)$ can be calculated by (16c) and (16d), respectively.

$$\begin{aligned} E_f = C_E\{e \mid e \in E \wedge [(\deg(e.s) \neq 1 \wedge \deg(e.t) = 1) \\ \vee (\deg(e.s) = 1 \wedge \deg(e.t) \neq 1)]\}, \end{aligned} \tag{16a}$$

```
Input: A digraph $D = \{V, E\}$
Output: set of coordinates of leaf vertices
1: for each $v \in V$ do
2:     leaf_number=0;
3:     for each $w \in V$ do
4:         if $w$ is neighbor of $v$ and $degree(w)$=1 then
5:             leaf_number=leaf_number+1;
6:         end if
7:     end for
8:     if leaf_number>0 then
9:         $i = l = 1$;
10:        Calculate $a_1$ by Eq.(13a), $L$ by Eq.(14) and $a_{rest\_at\_L}$ by Eq.(15);
11:        for each $w \in V$ do
12:            if $w$ is neighbor of $v$ and $w$ is a leaf vertex then
13:                $a = a_1 + (l - 1) * 9$;
14:                If $i > a$ then
15:                    $i = 1; l = l + 1; a = a + 9$;
16:                end if
17:                if $l = L$ then
18:                    $a = a_{rest\_at\_L}$;
19:                end if
20:                $r = r_v + (3l - 1)r_{leaf}$;// $r$: radius of the $l$th layer concentric circle. $r_v$: radius of $v$.
21:                $w_x = v_x + r * \cos(2\pi i/a)$;// $v_x$ and $w_x$: x-coordinates of vertices $v$ and $w$
22:                $w_y = v_y + r * \sin(2\pi i/a)$;// $v_y$ and $w_y$: y-coordinates of vertices $v$ and $w$
23:                $i = i + 1$;
24:            end if
25:        end for
26:    end if
27: end for
```

Algorithm 2: LD algorithm.

$$V_f = \{v \mid v \in V \wedge [\deg(v) > 1]\}, \tag{16b}$$

$$p_{v-force}(D) = \frac{|V_f|}{|V|}, \tag{16c}$$

$$p_{e-force}(D) = \frac{|E_f|}{|E|}, \tag{16d}$$

where deg(•) is degree of vertex •, *e.s* is source vertex of edge *e*, and *e.t* is target vertex of edge *e*.

Proposition 9. *Let $D = \{V, E\}$ be a digraph, with V the set of vertices and E the set of edges. Then $p_{leaf}(D) + p_{v-force}(D) = 1$.*

Proof. It is easy to get that $|V_f| = |V| - n_{leaf}(D)$ according to (16b). Then one can also get $p_{leaf}(D) = n_{leaf}(D)/|V|$ by (1) and $p_{v-force}(D) = |V_f|/|V|$ by (16c). Finally, $p_{leaf}(D) + p_{v-force}(D) = n_{leaf}(D)/|V| + |V_f|/|V| = (n_{leaf}(D) + |V_f|)/|V| = |V|/|V| = 1$. □

FDS algorithm applies forces only on non-leaf vertices. Note that cycle vertices always hold positions even though they are pulled or pushed by forces as cycle vertices must be non-leaf vertices. Finally, FDS algorithm is shown as Algorithm 3.

```
Input: A digraph $D = \{V, E\}$
Output: set of coordinates of vertices in $D$.
1: Step 1:Run HL-DPC algorithm on $D$;
2: Step 2:Run CD-CP algorithm on found cycles;
3: Step 3:Calculate force-subgraph $D_f$ from $D$;
4: Step 4:Run FR algorithm on $D_f$;
5: Step 5:Run LD algorithm on $D$;
```

Algorithm 3: FDS algorithm (a force-directed algorithm for drawing digraphs symmetrically).

The time complexity of FDS algorithm is contributed by all five steps. For every cycle vertex, its coordinate needs to be calculated only once in Step 2. For every star vertex, its coordinate also needs to be calculated only once in Step 5. The function of Step 3 is to find the force-subgraph according to the vertex degree. Thus, Steps 2, 3, and 5 are linear, and the worst time complexity of them is $O(|V|)$. According to [23], at best the time complexity of Step 1 (HL-DPC algorithm) would be $O(|V| + |E|)$; at worst it would be $O(|V| + |E| + \sum_{i=1}^{|D/D^3_{SCC}|} T_i(D^3_{SCC}))$. $T_i(D^3_{SCC})$ is given in (17). In (17), the direction parameter *ratio* is defined in Definition 4.

The time complexity of Step 4 (FR algorithm) would be $O(|V_f|^2)$. Thus, the time complexity of FDS algorithm would be between $O(|V| + |E| + |V_f|^2)$ and $O(|V| + |E| + |V_f|^2 + \sum_{i=1}^{|D/D^3_{SCC}|} T_i(D^3_{SCC}))$. For a digraph D, besides the size of D, practical time cost of FDS algorithm is also influenced by three indexes, the proportions of cycle, leaf, and non-leaf vertices.

$$T_i\left(D^3_{SCC}\right) = \begin{cases} \prod_{j=1}^{|V^3_{SCC}(i)|} S\deg^+\left(v_j\right) & if\ ratio\,(i) \le 0.6\ or\ ratio\,(i) > 1.6 \\ \prod_{j=1}^{|V^3_{SCC}(i)|} S\deg^-\left(v_j\right) & if\ 0.6 < ratio\,(i) \le 1.6 \end{cases} \quad (17)$$

5. Experimental Results

In total, 26 different relational datasets containing 6-50001 vertices are chosen to test related algorithms. 14 man-made datasets are used to test the proposed FDS algorithm when it is dominated by LD and CD-CP algorithms, respectively. 2 datasets are from the Gephi Chinese Tutorial [41], labeled as GCT; 1 dataset is collected by us in practical network, labeled as collected; 9 datasets are from the University of Florida Sparse Matrix Collection [42], labeled as sociology, biology, citation, sport, genealogy, response, and voting. Details of 26 datasets are listed in Table 3. The relational data used to support the findings of this study have been deposited in the Figshare repository (https://figshare.com/s/3cc5b3449b334aba2461) and are also available from the corresponding author upon request.

All algorithms are implemented in NetBeans IDE 8.0.1, under the Gephi 0.8.2 development kit. The configuration of experiments is operating system, WIN7, CPU, Intel(R) Core(TM) i5-4590 Quad CPU 3.30GHZ, Memory, 4G. In this section, three algorithms are chosen to compare to the FDS algorithm proposed in this paper, which are the classical $O(n^2)$ FR algorithm (FR) [24], the recent $O(n \log n)$ ForceAtlas2 algorithm (FA2) [29], and $O(n \log n)$ multilevel YifanHu algorithm (YFHu) [17].

About the parameter settings, firstly, the parameter δ is used in HL-DPC which is a subalgorithm of FDS algorithm. The parameter δ works only if there exists D^3_{scc} (SCC with 3 vertices at least) in digraphs. For the digraphs in Table 3, only 12 digraphs of them contain D^3_{scc} so that it is required to set parameter δ for them experimentally. The values of δ for the 12 datasets are listed in Table 4. The role of parameter δ is to provide the threshold for the procedure of heuristic DFS in HL-DPC algorithm. The threshold is calculated by $\delta \times |V^3_{scc}|$. If δ is set too large, then the threshold is larger than the number of vertices belonging to the cycle in D^3_{scc}, which would cause that the cycle can not be found. If δ is set too small, then the threshold is smaller than the number of vertices belonging to the cycle in D^3_{scc}, which might cause that the size of the found cycle is smaller than the size of the cycle in D^3_{scc}. In general, we need to set the value of δ for every D^3_{scc} because it is nearly impossible that there are two D^3_{scc} that are isomorphic. For example, there are two D^3_{scc} in the digraph of dataset Ooof-email, and we set δ to be 0.38 for first D^3_{scc} and 0.7 for second D^3_{scc}. In Table 4, if the value of δ is $*$, that means the value of δ has no impact on finding cycles of corresponding D^3_{scc}. In addition, FDS algorithm proposed in this paper is a force-directed method, and one part of FDS is FR algorithm. Thus some parameters related to force-directed algorithms also should be considered. In this paper, all algorithms are implemented in the software of Gephi 0.8.2. In order to compare FDS algorithm fairly to other algorithms, default parameter settings in Gephi 0.8.2 are used for FA2, YFHu, and FR algorithms. In particular, FDS takes same value of parameter "speed" as that of other two algorithms (FA2 and FR) because "speed" affects the efficiency of these force-directed algorithms.

16 drawings of 4 datasets (O, Stranke94, Cage3, and Cage4) are listed in Table 5, which are produced by FA2, YFHu, FR, and FDS algorithms, respectively. For the 4 datasets, all vertices belong to cycles. Thus the main goal of drawing these 4 datasets is to display cycles symmetrically. The symmetric measurement and time consumption of these 16 drawings are then listed in Table 6. In Table 6, optimal values of $E\sigma$, $D\sigma$, and time are bold conspicuously. According to human perception and the values of $E\sigma$ in Table 6, drawings produced by FDS are more symmetrical than those produced by other 3 algorithms. In addition, FDS spends less time than other 3 algorithms. In a word, FDS can produce more symmetrical drawings with less time consumption than other 3 algorithms for these 4 datasets.

12 datasets (Twitter07 and B, C, ..., L) are also used to test these four algorithms. There are only two kinds of vertices, *leaf* and *star vertices*, in these 12 datasets. Thus the main goal of drawing these 12 datasets is to display star-subgraphs symmetrically. 4 drawings of the typical dataset Twitter07 are listed in Table 7, which are produced by FA2, YFHu, FR, and FDS algorithms, respectively. The symmetric measurement and time consumption of these 4 drawings are listed in Table 7. In Table 7, optimal values of $E\sigma$, $D\sigma$, and time are bold conspicuously. No matter from the viewpoint of human intuition or according to the values of $E\sigma$, it can be seen that the drawing produced by FDS is most symmetrical. FDS spent so few time that can even be ignored.

Except for dataset Twitter07, changes of $E\sigma$ and time consumption of 43 drawings for other 11 datasets (B, C, ..., L) are shown in Figures 6(a) and 6(b), respectively. It can be seen that the drawings produced by FDS show their undoubted advantages on displaying symmetry ($E\sigma$) and efficiency (time). In a word, FDS can produce very symmetrical drawings within much less time than other 3 algorithms when datasets contain large proportion of *leaf vertices*.

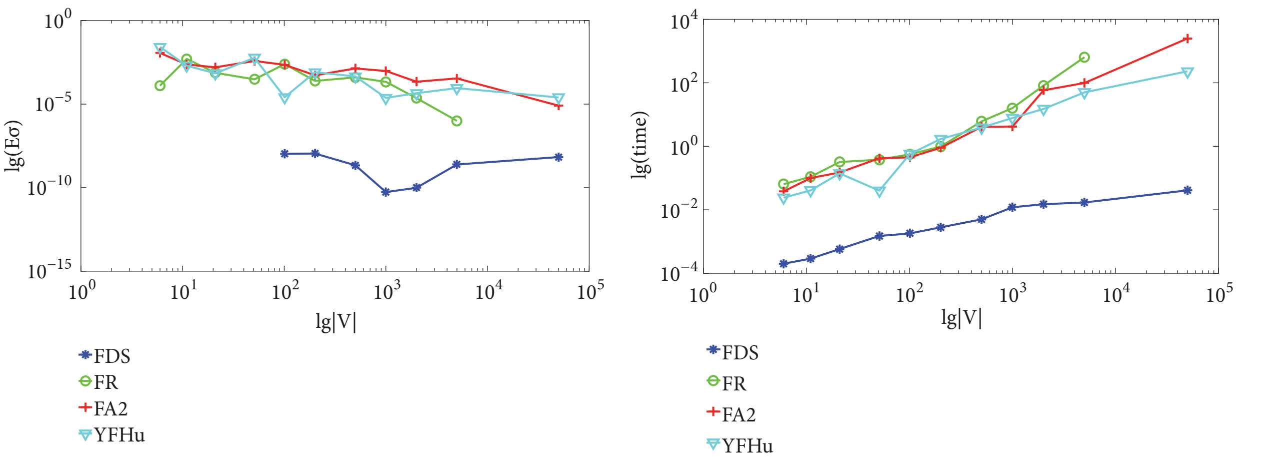
Unlike dataset Twitter07 drawn in Table 7 and datasets B-L involved in Figure 6, in many practical datasets, not

Table 3: Details of the 26 datasets used for algorithms test.

Datasets	type	$\lvert V\rvert$	$\lvert E\rvert$	Number of D^3_{scc}	$p_{leaf}(D)$	$p_{v-force}(D)$
B	Man-made	6	5	0	83.333%	16.667%
C	Man-made	11	10	0	90.909%	9.091%
D	Man-made	21	20	0	95.228%	4.772%
E	Man-made	51	50	0	98.039%	1.961%
F	Man-made	101	100	0	99.010%	0.990%
G	Man-made	201	200	0	99.502%	0.498%
H	Man-made	501	500	0	99.800%	0.200%
I	Man-made	1001	1000	0	99.900%	0.100%
J	Man-made	2001	2000	0	99.950%	0.050%
K	Man-made	5001	5000	0	99.980%	0.020%
L	Man-made	50001	50000	0	99.998%	0.002%
M	Man-made	5	7	1	0%	100%
N	Man-made	15	24	1	20%	80%
O	Man-made	30	30	3	0%	100%
Ooof-email	GCT	119	183	2	69.750%	30.250%
Twitter07	GCT	2666	2654	0	99.550%	0.450%
1	collected	278	302	0	82.734%	17.266%
Stranke94	sociology	10	90	1	0%	100%
Cage3	biology	5	19	1	0%	100%
Cage4	biology	9	49	1	0%	100%
Tina_DisCal	sociology	11	41	1	0%	100%
GD01_b	citation	18	37	1	0%	100%
Ragusa18	sociology	23	64	1	13.043%	86.957%
football	sport	35	118	0	8.571%	91.429%
EPA	response	4271	8965	10	47.577%	52.423%
EVA	genealogy	7252	6726	2	87.743%	12.257%

TABLE 4: Parameter setting of δ used in HL-DPC algorithm for datasets.

Datasets	Number of D^3_{scc}	Parameter δ	Number of cycles
M	1	$*$	1
N	1	0.6	1
O	3	$\{*, *, *\}$	3
Ooof-email	2	{0.38, 0.7}	2
Stranke94	1	0.91	1
Cage3	1	0.89	1
Cage4	1	0.89	1
Tina_DisCal	1	0.89	1
GD01_b	1	0.35	1
Ragusa18	1	0.75	1
EPA	10	$\{0.9, 0.7, 0.9, 0.7, 0.7, 0.4, *, 0.9, 0.4, 0.5\}$	9
EVA	2	$\{0.8, *\}$	1



(a) Changes of $E\sigma$ (b) Changes of time consumption

FIGURE 6: Changes of $E\sigma$ and time consumption of four algorithms on producing 43 drawings. Notes. (1) It is '43' not '44' because FR can not draw dataset L within bearable time. (2) In order to make Figures 6(a) and 6(b) more readable, the logarithm values of them are taken. Thus x axis stands for $\lg|V|$ and y axis $\lg(time)$ and $\lg(E\sigma)$. (3) $E\sigma$ of drawings produced by FDS for four datasets (B,C,D E) are invisible in Figure 6(a) because the values of them are 0.

all vertices belong to star-subgraphs. For dataset 1, for example, forces on many vertices have to be calculated although it contains many leaf vertices. 4 drawings produced by FA2, YFHu, FR, and FDS algorithms for dataset 1 are listed in Table 8. The symmetric measurement and time consumption of them are also listed. It can be seen that FDS provided best drawing for dataset 1 within shortest time. Compared to the drawing produced by FDS, the drawings produced by FA2 and YFHu did not display relationships between non-leaf vertices clearly enough although they displayed star-subgraphs symmetrically. And in the drawing produced by FR, star-subgraphs take up oversized area although relationships between vertices are displayed clearly.

In datasets M, Tina_DisCal, GD01_b, and Ragusa18, there are only a few vertices to which attractive and repulsive forces need to be calculated besides cycle vertices. Their drawings produced by the 4 algorithms are listed in Table 9. The symmetric measurement and time spent of the 16 drawings are listed in Table 10. It can be seen that the drawings produced by FDS have displayed cycles conspicuously and symmetrically. In addition, FDS algorithm provides performance comparable to $O(n \log n)$ YFHu algorithm [17] according to time consumption listed in Table 10.

In datasets N, Ooof-email, EPA, and EVA, there are both cycles and star-subgraphs. 16 drawings produced by 4 algorithms for the 4 datasets are listed in Table 11. The symmetric measurement and time spent of them are listed in

Table 5: 16 drawings produced by FA2, YFHu, FR, and FDS algorithms for 4 datasets (O, Stranke94, Cage3, and Cage4).

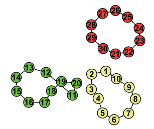
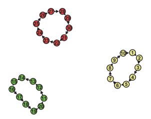
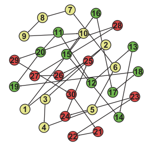
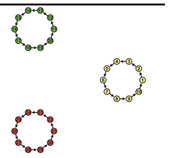
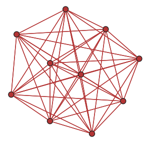
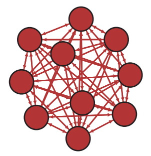
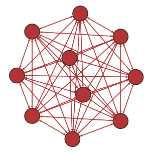
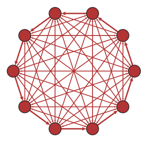
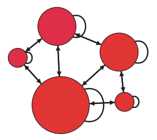
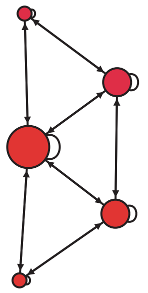
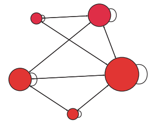
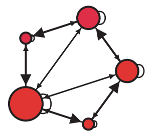
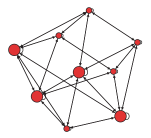
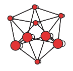
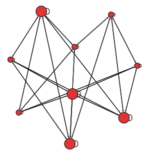
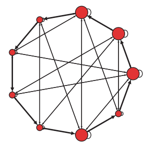
Datasets	FA2	YFHu	FR	FDS
O				
Stranke94				
Cage3				
Cage4				

Table 6: The symmetric measurement and time consumption (second) of 16 drawings in Table 5.

	Datasets	FA2	YFHu	FR	FDS
Eσ	O	7.89E-1	8.50E-1	8.56E-1	**7.73E-1**
	Stranke94	4.61E-2	2.51E-2	1.17E-1	**0**
	Cage3	3.24E-1	6.10E-1	3.17E-1	**3.16E-1**
	Cage4	4.61E-1	3.80E-1	1.91	**2.86E-1**
Dσ	O	3.21E-2	1.27E-2	7.18E-2	**2.41E-15**
	Stranke94	4.81E-35	1.20E-35	1.93E-34	**0**
	Cage3	**2.60E-3**	2.15E-1	4.05E-2	4.17E-2
	Cage4	4.32E-2	6.02E-2	1.87E1	**1.74E-2**
time	O	1.4	8.1E-2	1.6E-1	**3.8E-3**
	Stranke94	1.8E-1	1.5E-1	1.3E-1	**9.9E-3**
	Cage3	1.1E-1	3.4E-2	1.2E-1	**1.7E-3**
	Cage4	1.2E-1	7.9E-2	1.2E-1	**1.0E-3**

Table 7: 4 drawings produced by FA2, YFHu, FR, and FDS algorithms for dataset Twitter07. In addition, the symmetric measurement and time consumption (second) of them are also listed.

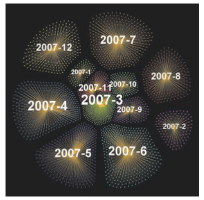
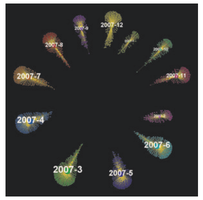
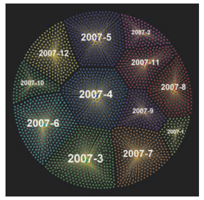
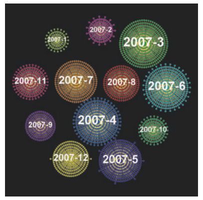
Dataset	FA2	YFHu	FR	FDS
Twitter07	 2007-12 2007-7 2007-1 2007-11 2007-10 2007-8 2007-4 2007-3 2007-9 2007-2 2007-5 2007-6	 2007-12 2007-8 2007-7 2007-11 2007-4 2007-6 2007-3 2007-5	 2007-5 2007-12 2007-11 2007-10 2007-8 2007-4 2007-9 2007-6 2007-3 2007-7	 2007-2 2007-3 2007-11 2007-7 2007-8 2007-6 2007-9 2007-4 2007-10 2007-12 2007-5
Eσ	3.65E-3	3.58E-3	5.06E-2	**8.43E-7**
Dσ	7.67E-3	4.64E-4	5.61	**1.90E-5**
time	6E1	1.3E1	7.4E1	**1.6E-1**

Table 8: 4 drawings produced by FA2, YFHu, FR, and FDS algorithms for dataset 1. In addition, the symmetric measurement and time consumption (second) of them are also listed.

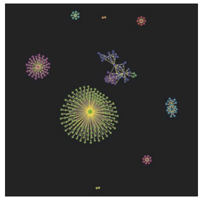
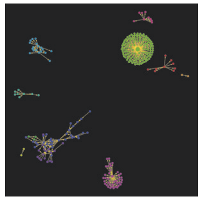
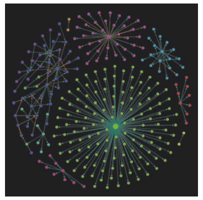
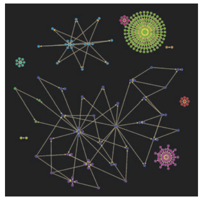
Dataset	FA2	YFHu	FR	FDS
1				
$E\sigma$	1.76E-1	3.01E-1	2.18E-1	**1.00E-1**
$D\sigma$	1.19	3.17	1.26	**8.81E-2**
time	7.7	7.4E-1	6.8E1	**3.4E-1**

TABLE 9: 16 drawings produced by FA2, YFHu, FR, and FDS algorithms for datasets M, Tina_DisCal, GD01_b, and Ragusa18.

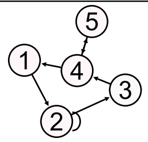
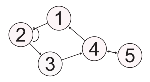
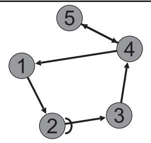
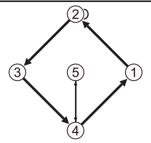
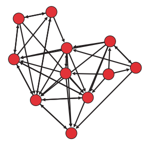
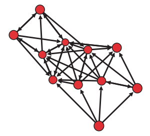
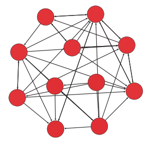
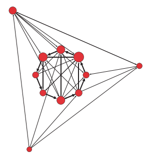
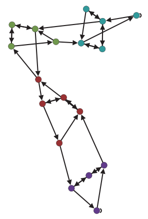
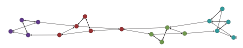
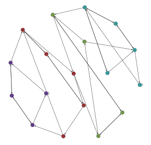
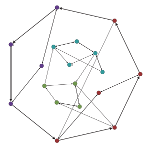
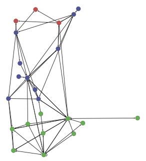
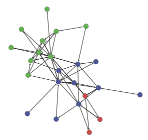
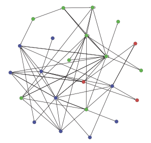
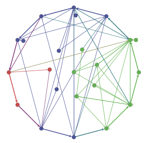
Datasets	FA2	YFHu	FR	FDS
M				
Tina_DisCal				
GD01_b				
Ragusa18				

Table 10: The symmetric measurement and time consumption (second) of 16 drawings in Table 9.

	Datasets	FA2	YFHu	FR	FDS
$E\sigma$	M	2.28E-1	8.50E-1	2.94E-1	**2.00E-1**
	Tina_DisCal	4.34E-1	1.04	1.17E-1	**4.08E-1**
	GD01_b	8.70E-1	5.37	7.80E-1	**7.45E-1**
	Ragusa18	5.79E-1	5.74E-1	5.14E-1	**5.00E-1**
$D\sigma$	M	1.70E-2	7.47E-2	4.28E-2	**1.67E-2**
	Tina_DisCal	1.17E-1	6.31E-1	1.53E-1	**7.81E-2**
	GD01_b	5.92E-1	1.28E2	**9.11E-2**	3.34E-1
	Ragusa18	2.10E-1	4.82E-1	**7.81E-2**	2.77E-1
time	M	7.9E-2	**2.7E-2**	7.9E-2	6.2E-2
	Tina_DisCal	1.6E-1	2.5E-2	1.7E-1	**1.1E-1**
	GD01_b	4E-1	**1.1E-1**	1.4E-1	**1.1E-1**
	Ragusa18	1.8E-1	1.6E-1	1.7E-1	**1.5E-1**

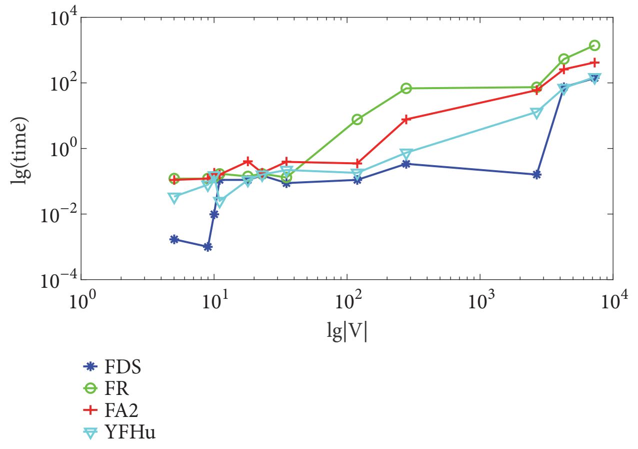


Figure 7: Comparison of time consumption by 4 algorithms on 12 practical datasets.

Table 12. It can be seen that cycles in the 4 datasets are drawn symmetrically and conspicuously by FDS. As the directed edges are displayed as straight line, thus cycles are displayed as regular polygons. For example, there are two cycles in dataset Ooof-email and one cycle in dataset EVA, which are displayed as the regular triangle, the regular heptagon, and the square in corresponding drawings of Table 11. Compared to other 3 algorithms, star-subgraphs in the 4 datasets are drawn symmetrically by FDS with taking moderate-size area. The values of $E\sigma$ listed in Table 12 also show the advantage of FDS on display of symmetry. According to time consumption recorded in Table 12, FDS can produce symmetrical drawings with taking comparable time to YFHu algorithm.

There may be a concern that FDS algorithm would force nonsymmetric digraphs into symmetric drawings. The experiments on dataset football (Table 13) can be used to clarify it. There is neither cycles nor star-subgraphs in dataset football so no much symmetry is needed to be drawn. In contrast to earlier experiments, it is the drawing produced by FR not the drawing produced by FDS which achieves the best value of $E\sigma$. Furthermore, the drawing produced by FDS spends least time maybe because there are 3 leaf vertices on which FDS does not have to calculate forces.

Furthermore, the comparisons of time consumption of these 4 algorithms on 12 practical datasets (Cage3, Cage4, Stranke94, Tina_DisCal, GD01_b, Ragusa18, football, Ooof-email, 1, Twitter07, EPA, and EVA) are also shown in Figure 7. It can be seen that FDS (blue line) provides performance comparable to YFHu (cyan line) algorithm with time complexity $O(n \log n)$.

6. Conclusion and Future Work

In this paper, an objective symmetric metric is proposed based on vertex coordinate calculation to measure the extent to which drawings can be considered symmetric.

TABLE 11: 16 drawings produced by FA2, YFHu, FR, and FDS algorithms for datasets N, Ooof-email, EPA, and EVA.

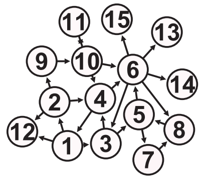
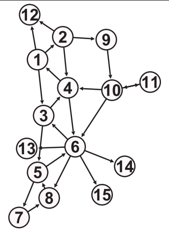
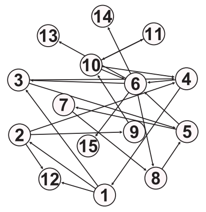
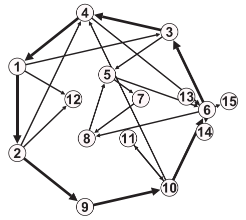
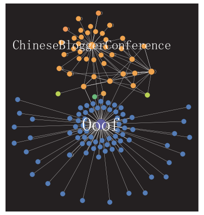
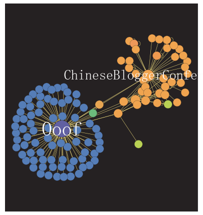
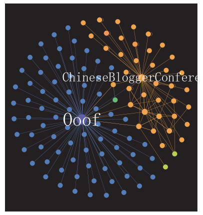
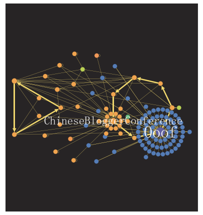
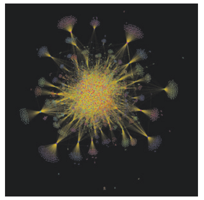
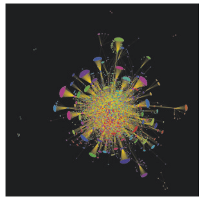
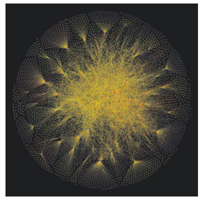
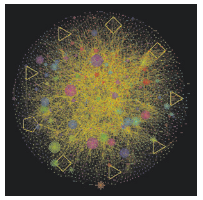
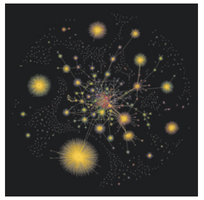
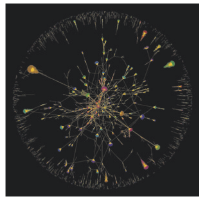
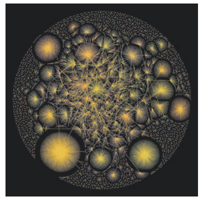
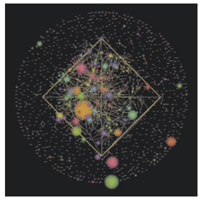
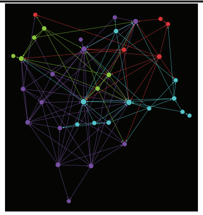
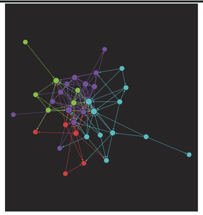
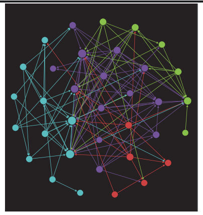
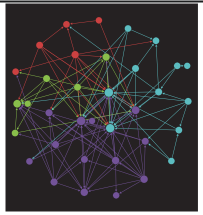
Datasets	FA2	YFHu	FR	FDS
N				
Ooof-email				
EPA				
EVA				

Table 12: The symmetric measurement and time consumption (second) of 16 drawings in Table 11.

	Datasets	FA2	YFHu	FR	FDS
$E\sigma$	N	2.87E-1	3.97E-1	2.62E-1	**2.61E-1**
	Ooof-email	5.18E-1	1.45E-1	3.10E-1	**1.15E-1**
	EPA	8.33E-1	9.03E-1	6.30E-1	**6.00E-1**
	EVA	5.68E-1	6.32E-1	4.05E-1	**2.26E-1**
$D\sigma$	N	1.32E-1	3.95E-1	5.84E-2	**5.45E-2**
	Ooof-email	1.79E1	2.67E-1	2.69	**1.73E-1**
	EPA	2.87E1	2.75E2	**5.77**	1.07E1
	EVA	6.93E1	7.12E2	1.24E2	**7.82**
time	N	6.9E-1	**7.6E-2**	1.1E-1	8.7E-2
	Ooof-email	3.5E-1	1.8E-1	7.6	**1.1E-1**
	EPA	2.6E2	**7E1**	5.3E2	7.4E1
	EVA	4.2E2	1.5E2	1.4E3	**1.4E2**

Table 13: 4 drawings produced by FA2, YFHu, FR, and FDS algorithms for dataset football. In addition, the symmetric measurement and time consumption (second) of them are also listed.

Dataset	FA2	YFHu	FR	FDS
football				
$E\sigma$	5.64E-1	6.70E-1	**4.56E-1**	4.62E-1
$D\sigma$	5.64E-1	8.22E-1	3.00E-1	**2.38E-1**
time	3.9E-1	2.2E-1	1.3E-1	**8.8E-2**

The results of measurement by the proposed metric are consistent with human intuition. In addition, the metric can measure symmetries rapidly because it does not involve automorphisms which is computational hardly. Furthermore, the metric is proven theoretically and experimentally to be robust for rotation and scaled change of drawings. And the robustness conforms to human perception of symmetry.

In order to display star-subgraphs and cycles symmetrically, a force-directed algorithm called FDS is proposed. The ability of FDS algorithm to display symmetry is validated by several groups of experimental tests. FDS algorithm's time consumption is comparable to YFHu algorithm with time complexity $O(n \log n)$. In some special datasets of which displaying of star-subgraphs is main goal, FDS spends much less time than YFHu algorithm. Note that FDS algorithm can only perform its advantage to the digraphs containing star-subgraphs and cycles because FDS algorithm is committed to display the symmetry of star-subgraphs and cycles in drawings. If one applies FDS algorithm to the digraphs which do not contain star-subgraphs and cycles, FDS algorithm would degenerate into FR algorithm without negative impact, which means FDS algorithm would not force nonsymmetric digraphs into symmetric drawings. Corresponding experimental results also verified this conclusion. Thus, the applicable conditions of FDS algorithm are the existence of star-subgraphs or cycles in digraphs.

In future, we may try to use other methods (e.g., distribution entropy) to construct another objective metric on symmetry. Although the FDS algorithm proposed in this paper only focus on drawing two structures, cycles and star-subgraphs, symmetrically, FDS algorithm has impossible impact. It can provide reference and guidance for designing other similar methods for drawing other graph structures, having inherent symmetry, symmetrically. Besides, it should be noted that FR algorithm is used in FDS algorithm because its good performance on aesthetics balance, though it requires $O(n^2)$ calculations. One may speed up FDS algorithm by replacing FR with other methods without sacrifice of aesthetics.

Data Availability

The relational data used to support the findings of this study have been deposited in the Figshare repository (https://figshare.com/s/3cc5b3449b334aba2461) and are also available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank LiuYong for providing two datasets Ooof-email and Twitter07, Wang Fei for the help on programme. This work was supported by the Qian Education Cooperation (KY[2018] No. 318), Qian Science and Technology Cooperation (LH[2015] No. 7011), and Chongqing Research Program of Basic Research and Frontier Technology (cstc2017jcyjAX0144).

References

[1] M. Ghoniem, J.-D. Fekete, and P. Castagliola, "A comparison of the readability of graphs using node-link and matrix-based representations," in *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium*, pp. 17–24, 2005.

[2] S. Bhanji, H. C. Purchase, R. F. Cohen, and M. James, "Validating graph drawing aesthetics: A pilot study," Technical Report, University of Queensland Department of Computer Science, 1995.

[3] H. C. Purchase, "Metrics for graph drawing aesthetics," *Journal of Visual Languages and Computing*, vol. 13, no. 5, pp. 501–516, 2002.

[4] C. Ware, H. Purchase, L. Colpoys, and M. McGill, "Cognitive measurements of graph aesthetics," *Information Visualization*, vol. 1, no. 2, pp. 103–110, 2002.

[5] R. J. Lipton, S. C. North, and J. S. Sandberg, "A method for drawing graphs," in *Proceedings of the the first annual symposium*, pp. 153–160, Baltimore, Maryland, United States, June 1985.

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

[7] G. D. Battista, R. Tamassia, and I. G. Tollis, "Area requirement and symmetry display of planar upward drawings," *Discrete & Computational Geometry*, vol. 7, no. 1, pp. 381–401, 1992.

[8] H. de Fraysseix, "An heuristic for graph symmetry detection," in *International Symposium on Graph Drawing*, Lecture Notes in Comput. Sci., pp. 276–285, Springer, 1999.

[9] D. Abelson, S.-H. Hong, and D. E. Taylor, "A group-theoretic method for drawing graphs symmetrically," in *Graph Drawing Software*, vol. 2528 of *Lecture Notes in Comput. Sci.*, pp. 86–97, Springer, Berlin, 2002.

[10] S.-H. Hong and H. Nagamochi, "A linear-time algorithm for symmetric convex drawings of internally triconnected plane graphs," *Algorithmica. An International Journal in Computer Science*, vol. 58, no. 2, pp. 433–460, 2010.

[11] F. Van Ham and B. E. Rogowitz, "Perceptual organization in user-generated graph layouts," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1333–1339, 2008.

[12] H. C. Purchase, B. Plimmer, R. Baker, and C. Pilcher, "Graph drawing aesthetics in user-sketched graph layouts," in *Proceedings of the Eleventh Australasian Conference on User Interface*, vol. 106, pp. 80–88, Australian Computer Society, 2010.

[13] H. C. Purchase, C. Pilcher, and B. Plimmer, "Graph drawing aesthetics∩┐∩┐created by users, not algorithms," *IEEE Transactions on Visualization Computer Graphics*, vol. 18, no. 1, p. 81, 2012.

[14] C.-C. Lin, W. Huang, W.-Y. Liu, S. Tanizar, and S.-Y. Jhong, "Evaluating esthetics for user-sketched layouts of clustered graphs with known clustering information," *Journal of Visual Languages and Computing*, vol. 37, pp. 1–11, 2016.

[15] M. Y. Becker and I. Rojas, "A graph layout algorithm for drawing metabolic pathways," *Bioinformatics*, vol. 17, no. 5, pp. 461–467, 2001.

[16] S. Hachul and M. Junger, "Drawing large graphs with a potential field-based multi-level algorithm," *Graph Drawing Software*, pp. 285–295, 2004.

[17] Y. Hu, "Efficient, high-quality force-directed graph drawing," *Mathematica Journal*, vol. 10, no. 1, pp. 37–71, 2005.

[18] S. Tee Teoh and M. Kwan-Liu, "RINGS: A Technique for Visualizing Large Hierarchies," in *Graph Drawing Software*, vol. 2528 of *Lecture Notes in Computer Science*, pp. 268–275, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[19] E. R. Gansner and Y. Koren, "Improved circular layouts," in *Graph Drawing Software*, vol. 4372 of *Lecture Notes in Comput. Sci.*, pp. 386–398, Springer, Berlin, 2007.

[20] E. Di Giacomo, W. Didimo, G. Liotta, and H. Meijer, "Computing radial drawings on the minimum number of circles," *Journal of Graph Algorithms and Applications*, vol. 9, no. 3, pp. 365–389, 2005.

[21] F. Van Ham and M. Wattenberg, "Centrality based visualization of small world graphs," *Computer Graphics Forum*, vol. 27, no. 3, pp. 975–982, 2010.

[22] D. Archambault, T. Munzner, and D. Auber, "Smashing peacocks further: drawing quasi-trees from biconnected components," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 813–820, 2006.

[23] T. Xu and Q. Zhang, "Research on suspicious transaction recognition based on heuristic listing of directed primary circuit," *Journal of Nanjing University(Natural Sciences*, vol. 52, no. 5, p. 879, 2016.

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

[25] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information Processing Letters*, vol. 1, no. 4, pp. 132-133, 1972.

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

[27] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7–15, 1989.

[28] N. R. Quinn and M. A. Breuer, "A Forced Directed Component Placement Procedure for Printed Circuit Boards," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 26, no. 6, pp. 377–388, 1979.

[29] 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, Article ID e98679, 2014.

[30] 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.

[31] D. Harel and Y. Koren, "A fast multi-scale method for drawing large graphs," in *Graph drawing*, pp. 183–196, Springer, 2001.

[32] P. Gajer and S. G. Kobourov, "GRIP: Graph drawing with intelligent placement," *Journal of Graph Algorithms and Applications*, vol. 6, no. 3, pp. 203–224, 2002.

[33] C. Walshaw, "A multilevel algorithm for force-directed graph-drawing," *Journal of Graph Algorithms and Applications*, vol. 7, no. 3, pp. 253–285, 2003.

[34] J. Barnes and P. Hut, "A hierarchical *O(N log N)* force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, 1986.

[35] E. R. Gansner, Y. Koren, and S. North, "Graph drawing by stress majorization," in *Proceedings of the 12th International Symposium on Graph Drawing, GD 2004*, pp. 239–250, USA, October 2004.

[36] V. De Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," *Advances in Neural Information Processing Systems*, vol. 15, pp. 1959–1966, 2003.

[37] U. Brandes and C. Pich, "Eigensolver methods for progressive multidimensional scaling of large data," in *Graph Drawing*, pp. 42–53, Springer, 2006.

[38] E. R. Gansner, Y. Hu, and S. North, "A maxent-stress model for graph layout," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 6, pp. 927–940, 2013.

[39] K. Rosen, *Discrete Mathematics and Its Applications*, McGraw-Hill Science, 7th edition, 2011.

[40] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[41] L. Ooof, "Gephi chinese tutorial," 2012, https://www.udemy.com/gephi/.

[42] V. Batagelj and A. Mrvar, "Pajek datasets," 2006, http://vlado.fmf.uni-lj.si/pub/networks/data/.