Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080145

Scalable Visualization of Semantic Nets using Power-Law Graphs

Ajaz Hussain[1], *Khalid Latif*[1,*], *Aimal Tariq Rextin*[1], *Amir Hayat*[2] *and Masoon Alam*[2]

[1] School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Sector H-12, Islamabad 44000, Pakistan
[2] Comsats Intitute of Information Technology, Islamabad 44000, Pakistan

Received: 1 Jul. 2013, Revised: 10 Nov. 2013, Accepted: 13 Nov. 2013
Published online: 1 Jan. 2014

Abstract: Scalability and performance implications of semantic net visualization techniques are open research challenges. This paper focuses on developing a visualization technique that mitigates these challenges. We present a novel approach that exploits the underlying concept of power-law degree distribution as many realistic semantic nets seems to possess a power law degree distribution and present a small world phenomenon. The core concept is to partition the node set of a graph into power and non-power nodes and to apply a modified force-directed method that emphasizes the power nodes which results in establishing local neighborhood clusters among power nodes. We also made refinements in conventional force-directed method by tuning the temperature cooling mechanism in order to resolve 'local-minima' problem. To avoid cluttered view, we applied *semantic filtration* on nodes, ensuring zero loss of semantics. Results show that our technique handles very large scale semantic nets with a substantial performance improvement while producing aesthetically pleasant layouts. A visualization tool, *NavigOWL*, is developed by using this technique which has been ported as a plug-in for Protege, a famous ontology editor.

Keywords: Ontology visualization, semantic net visualization, power law graphs, scalable directed graphs, force-directed graphs.

1 Introduction

Due to recent advancements in semantic web, the web of data is continuously growing. One evidence is *Linked Data* initiative[1], which has resulted into billions of triples. Similarly, several large ontology structures have evolved over the last few years. There is a growing need of effective visualization methods that could be adopted for an effective representation of ontologies in order to fully understand the structures.

Literature indicates that several research initiatives have targeted 2D and 3D ontology visualization techniques [16,7,3,29,22,14,6,8,9,23,17,10,31,11,13,19]. Several tools exist to visualize the semantic nets [20]. Different layout techniques have been implemented in these tools, but each technique has shortcomings due to which either the structure of ontology becomes ambiguous to the user or the view losses the semantics. As the ontology size increases, several issues such as *node cluttering*, *edge crossings*, *local minima problem*, *angular resolution problem*, *computational inefficiency*, *shape/view distortion*, *ineffective rendering techniques*, and *asymmetrical drawings*, are observed. Hence an effective and scalable visualization technique is needed, that should clearly depict the structure of complete ontology while maintaining persistent aesthetic constraints. We present the optimized, scalable, and performance oriented directed graph layout technique to visualize the large-scale ontology graph by maintaining the aesthetics aspects of visualization as discussed in [25].

We exploited the underlying concept of power-law degree distribution topology pertaining to the property that a small proportion of nodes have a high degree (power nodes) while the vast majority of nodes have a low degree (non-power nodes). Thus, in a realistic network, there exists few power-nodes that needs to be emphasized more as compared to other non-power nodes. The basic idea is to partition all the nodes/vertices based upon power-law degree distribution and then to apply our

[1] http://linkeddata.org

* Corresponding author e-mail: khalid.latif@seecs.nust.edu.pk

optimized and modified force-directed layout algorithm which manages scalability and performance implications.

We begin by summarizing previous approaches that have been applied to graph layout and their aesthetic measures along with review of various tools of semantic net visualizations in Section 2. Section 3 discusses the detailed design of our novel graph layout technique along with refinements. In Section 4 we discuss our developed visualization tool, *NavigOWL*, and plug-in for Protege. In Section 5, we demonstrate how our algorithm achieves substantially better performance and mitigates scalability challenge by comparing it to the existing approaches while visualizing large scale semantic nets.

2 Related Work

In this section we discuss the generally accepted aesthetic criteria for graph layouts and describe various graph drawing layout techniques that have been implemented so far. Further, we discuss few existing ontology visualization tools.

2.1 Graph Drawing Aesthetics and Layouts

There are certain graph drawing constraints that a visualization must follow. These constraints are applied on a graph in order to get better understanding and structure of graph. These constraints are aesthetically measured. We are concerned with an optimized and scalable layout technique that posses generally accepted aesthetic criteria [25].

Moreover, visualization of semantic net gets complex due to its structure or role-relations hierarchy defined in an ontology. One node can be linked to many other nodes, so the user is hardly able to understand the structure of semantic net. In order to create clear mental map for the user, a layout has to be applied to the visualization. Following are few existing layout techniques that are discussed in detail.

The most popular layout algorithms are based on force-directed or spring-embedded methods [8,10,9,13, 29,4,17]. Eades designed the earliest spring-embedded model for graph drawing [8]. The basic idea was to embed a graph by replacing the vertices by steel rings and replace each edge with a spring to form a mechanical system. The vertices are placed in some initial layout and then released so that the spring forces on the rings move the system to a minimal energy state. The attractive force is calculated between neighbors with complexity of $\Theta(|E|)$ whereas the repulsive force is calculated between every pair of vertices with complexity of $\Theta\left(|V^2|\right)$. This technique generates aesthetically pleasing layouts. However, the repulsive force complexity poses serious computational issues for large graphs, hence it has to be limited to few hundred nodes. Moreover, this technique highlights certain shortcomings like *occluding vertices with edges*, *edge-crossings*, and *angular resolution problem* i.e. the angle between incident edges may be too small, and it may get smaller in case of larger graphs.

Another variant by Fruchterman and Reingold [13] simplified forces formulae and made several refinements by using *cooling schedule* to limit nodes maximum displacement. Repulsive force is still being calculated between every node in a graph, yielding to a complexity of $\Theta\left(|V^2|\right)$.

Similarly, a modified spring layout was presented by Huang et al. [17], in which they used *Online Force-Directed Animated Visualization*(OFDAV) technique for assisting web-navigation. This technique describes, which most of existing visualization systems have problems presenting, huge graphs like fish-eye [26], hyperbolic browser, cone trees [23]. The main issue of this technique is *zero angular resolution problem* i.e. the smallest angle between two neighboring edges incident on common vertex. Also, no role- relation hierarchy and context details are observed in graphs and this technique also does not resolve issue of overlapping edges of common vertex.

Later Lin and Yen also proposed a variant of Eades [10]. As many conventional force-directed methods are based on either vertex-vertex repulsion [8,13] or vertex-edge repulsion [17,29], therefore this approach is an enhancement which is based upon *edge-edge repulsion* to draw graphs. Although this technique resolved problem of *minimum angular resolution* [12], but it was not scalable and results show the drawings of smaller graphs with upto few dozen nodes. Further, this technique does not always guarantee to produce nice symmetrical drawing and poses *local minima problem* where forces get too week to spread graph. The main issue of this algorithm is its complexity which is in square. i.e. $\Theta\left(|N|^2+|E|\right)$. Hence performance of algorithm is worst in case of large scale ontologies.

From the discussion it is evident that despite various modifications and implementations, the force-directed algorithm is not scalable and gives worst performance in case of large ontologies. There is a need to optimize it for managing scalability challenge and to preserve aesthetic visualization.

2.2 Graph Clustering Algorithms

Apart from traditional force directed algorithms, significant research work has been done on clustering of the graphs [30,9,1,15,4,24,27,32]. Cluster graph is another important concept to visualize semantic nets. The clusters are built for the aesthetic view of graph structure. Few implementations, by tuning the force directed algorithms with clustering techniques, are given in [4,9, 27]. In this section, we review various approaches being adopted to build the graph clusters.

Clusters or groups are investivated for biological and social community structures by Girvan [15] be exploiting centrality in order to find community boundaries. The worst case complexity of algorithm though is $\Theta(m^2n)$ where m represents the number of edges and n as number of vertices. This technique was not highly scalable and supported few thousands nodes. Moreover, it is algorithmically complex as it operates on edges. In case of sparse graphs, its complexity is $\Theta(n^3)$, which is inefficient.

Another variant of clustering approach was presented by Yuruk et al. [32] in which finding the hierarchical structure of clusters without any input parameters was discussed. They presented, a *hierarchical structural clustering* algorithm for networks. Although, this approach is considered as highly effective in finding hierarchical clusters in social networks, but the implementations showed small scale graphs, thus this approach is also not highly scalable. Algorithm is not efficiently applicable in semantic networks, as using only structural similarity is not sufficient for building clusters. Results also showed node cluttering and edge-overlapping issues.

Wallner also described hierarchical cluster graphs that impose higher level of granularity controlled by users [27]. Although cluster center detection technique is efficient and controlled by user-defined threshold, however, this technique has certain shortcomings. Algorithm's complexity is very high due to cluster build phase and further Fruchterman and Reingold force-directed algorithm [13] is applied, which has high complexity. Because of breakdown condition problem, in some cases, algorithm aborts too early, hence shows long distances between meta-nodes. Finally, no computation time and scalability aspects are mentioned in literature.

2.3 Power Law Graphs

Several research efforts have targeted the power-law graphs and their combinations with the traditional force-directed algorithms [16] by exploiting the underlying structure of power-law network distribution [6,2,5]. In this section we review the research work related to power-law graphs.

Chan and colleagues presented a novel algorithm of *Out Degree Layout* for the visualization of large scale network topologies [6]. They divided the whole network into multiple layers based upon the Out Degree (the number of edges coming out of the node). They adopted the *Power Law topology*, which is commonly being found in topology networks. *A power-law topology has the property that a small proportion of nodes have a high out-degree i.e., have many connections to other nodes, while the vast majority of nodes have a low out-degree, i.e., have connections to few nodes* [6]. The methodology adopted is to layout the nodes with highest out-degree first, then to layout the nodes with smaller out-degrees and so on. They partitioned the graph into multiple layers based upon out-degrees. The graph layout is being adjust by using *Fruchterman-Reingold (FR) Force Directed Algorithm* [13]. Complexity of FR algorithm is $\Theta\left(|N|^2+|E|\right)$, but the overall complexity of their algorithm is $\Theta\left(\sum|N_k|^2+\sum|E_k|\right)$, where N_k and E_k presents the No. of nodes and edges in layer L_k respectively. They tested the BGP networks of up to 7,000 nodes and compared the results with traditional algorithms (including [10,13,28]) and achieved significant improvement in aesthetic layouts, though the view gets cluttered if the structure of network is complex.

Focus-based filtering and clustering technique in power-law network graphs provides better layout as compared to classical filtering technique as it is based upon power-law distribution [5]. Dense graphs can be filtered into clear view. However, scalability remains an open challenge as this technique is only demonstrated for small graph of 1,511 nodes and 7,902 co-authoring links. Moreover, it is computationally inefficient as firstly cluster cores are extracted and then layout is applied. Also, changing user-focus in this technique, gives cluttered view in few silhouettes.

In power-law topology a small portion of nodes have many connections to other nodes which means that in a semantic net, core nodes have the maximum degree and in the whole graph, there exists very few core-nodes. Similarly, as the degree of the nodes increase, the frequency behavior of the degree decreases, which is the basic property of mathematical power-law. It has been proven to be effective by various approaches that the interaction networks (Graphs containing actors and relationships among actors) or semantic nets exhibit the power-law behavior. Thus we can follow the power-law property in ontology visualization as ontology also exhibit power-law property, as the degree of nodes increases, its frequency decreases (i.e frequency of core nodes is very low) [5]. Our technique also exploit the underlying concept of power-law degree distribution, that we discuss in Section 3.

2.4 Ontology Visualization

Various ontology visualization tools are in use including *Cytoscape*, *Giny*, *graphViz*, *HyperGraph*, *rdfGravity*, *IsaViz*, *Jambalaya*, *Owl2Prefuse*, and *SocNetV* [3,19]. Most of these tools either lack in visualizing role-relation hierarchies of complete ontology or in aspect of drawing layouts. Figure 1 represents the visualization of *Amino Acid Ontology* of 1,484 triples, the drawing layout is fine but it misses role-relation hierarchy and simply represents classes along with their instances. Moreover, the animation doesn't stops in layout process as graph continuously adjusts itself which makes it harder for user to preserve mental map.

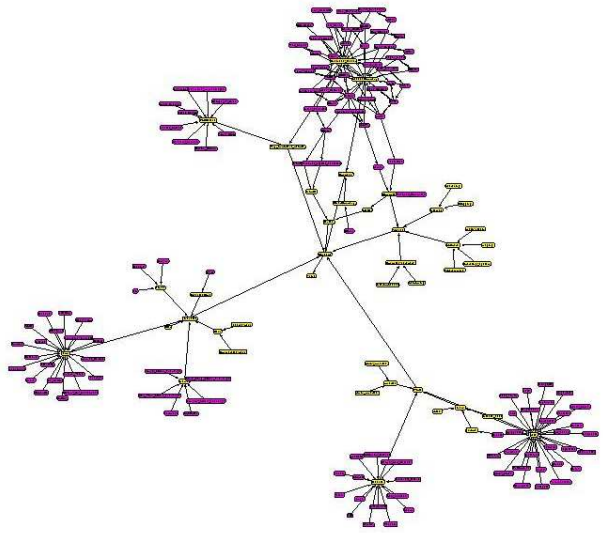


Fig. 1: Owl2Prefuse: view of classes and their instances, missing role-relation hierarchy, and other constructs in AminoAcid ontology graph

We also visualized same *Amino Acid* ontology to compare its graph in another popular ontology visualization tool named as *SocNetV*. The tool is enriched with various graph properties and used mostly to visualize and exploit the properties of social networks. The tool is computationally not very efficient, layouts are ambiguous, and lacks role-relation hierarchy. Figure 2 represents snapshot of amino-acid ontology after applying force-directed layout on, it shows ambiguous view with lack of details about role-relations.

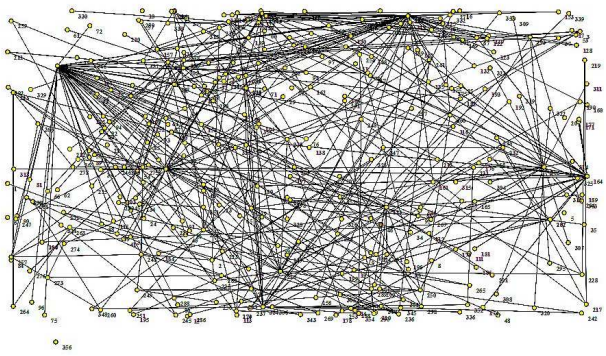


Fig. 2: SocNetV: Force-directed layout applied over AminoAcid ontology, cluttered view with node-overlaps, missing role-relations details.

rdfGravity is another popular ontology visualization tool. It represents the ontology graph with complete role-relation hierarchy and in full context along with literals and other constructs with filtering features. Unfortunately, it does not support any graph drawing layout so it gives holistic view after filtering but gives cluttered view for complete ontology graph. Figure 3 represents graph of *TransOntology-Bhakti*, this ontology graph consists of only 195 triples yet we can observe cluttered view and node-overlap in asymmetrical view of ontology graph. Its also not scalable to support large ontology graphs as the view becomes ambiguous and unstructured.

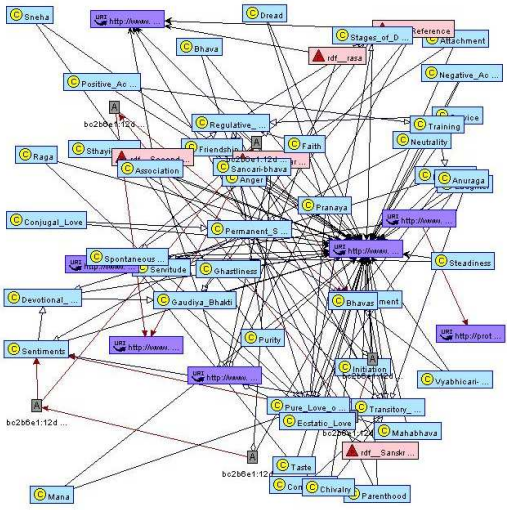


Fig. 3: rdfGravity: Cluttered view with node-overlap, asymmetrical drawing on small TransOntology-Bhakti graph.

Current ontology visualization tools tend to avoid scalability issues by limiting the number of visible nodes on graph canvas to about 10,000. *OntSphere* reports occlusion and label-overlap problems for little over 10,000 nodes. Another problem in visualization tools over large scale ontologies is node labels display, similarly visualization of relation links is also problematic. Figure 4 shows snapshot of *OntoGraf*, where we can observe cluttered view and node-overlap over ontology of just 195 triples.

Similarly, use of *TGVizTab* and *OntoViz* is not possible when relation links are visible even for an ontology of less than 300 nodes [19]. In *Jambalaya*, users cannot exploit the relation links. [18] categorized the existing ontology visualization tools that support up to 10,000 nodes. Our main challenge is to cope with scalability issue as large scale ontologies contains million of triples.

Summarizing the discussion on existing visualization tools, following shortcomings are observed which needs

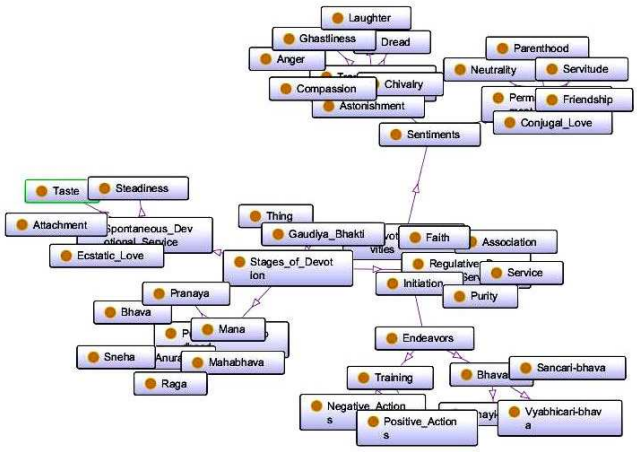


Fig. 4: OntoGraf spring layout: Cluttered view and node-overlap

to be covered in order to focus on ontology visualization domain.

- –Many visualization tools support graphs up to few hundred nodes only like rdfGravity, Jambalaya, GraphViz.
- –In large scale graphs these tools either take significant time in computation and/or produce ambiguous layouts as being observed in rdfGravity.
- –Node cluttering and edge overlap issues are also present as in Prefuse, graphViz, and OntGraf.
- –Tools are not enriched enough for describing role-relation hierarchy, like in OntGraf, a Protege plugin, it only visualizes class-hierarchy.
- –Force-directed and spring layouts are implemented in several visualization tools, however, local-minima problem has been found in case of large scale ontologies as observed in *SocNetV*.

3 Design and Methodology

This section covers the design of our proposed ontology algorithm along with sub-functions detail. Complexity of the algorithm has been discussed and compared with other approaches as well.

3.1 Power Law Based Ontology Visualization Algorithm

The proposed layout algorithm exhibits the underlying structure of mathematical power-law property along with existing force-directed algorithm. The outline of core algorithm is as follows:

1.Sort nodes as per degrees and convert semantic net to bipartite graph of Power and Non Power nodes.
2.While $temperature \neq 0$
- –Calculate attraction force among Power-Nodes and their neighboring nodes.
- –Calculate repulsive force among Power-Nodes.
- –Calculate attraction force among Non-power Nodes and their neighboring Nodes.
- –Calculate repulsive force among Non-power Nodes.
- –Calculate nodes positions and update the (x; y) coordinates of each node.
- –Reduce *temperature* at each iteration.

The core algorithm contains various methods which are discussed here in detail.

3.1.1 NodeDegreeMapping Method

This method is central to the proposed power law based algorithm. The basic idea of this method is to partition the nodes of a graph into power and non-power nodes based upon their degree distribution. We also tuned this method to apply variant node-scale on graph canvas based upon degree distribution (i.e the node with maximum scale on canvas will be the node with maximum degree). By doing so, visualization can provide aesthetic aspect by displaying nodes of variant scale on graph canvas. We defined scaling factor of node by following equation.

$$\sigma_i = \left[\frac{d_i}{\Delta(G)}\right] \times \kappa \tag{1}$$

where,

$\sigma_i =$ scale of node i.
$d_i =$ degree of node i.
$\Delta(G) =$ maximum degree of graph G.
and $\sigma_i \leq \kappa$; where κ is a defined constant.

The outline of this method is as follows:

1.Sort node $\leftrightarrow$ degree distribution of whole graph.
2.Extract power nodes, i.e. top 20% nodes from the sorted distribution.
3.Extract non power nodes, i.e. remaining 80% nodes.
4.Calculate each node's scale relative to its degree as per Equation 1.

3.1.2 AttractionForce Method

This method is based on the force-directed model. The basic principle of this method is to bring together the nodes which are connected by an edge, than acts like a spring between two nodes. The aim is to bring all neighboring nodes close to their power nodes, to build a local cluster around power nodes. An attraction force is

Algorithm 1: AttractionForce

Data: n $\longrightarrow$ *node*; d $\longrightarrow$ *degree*;
N $\longrightarrow$ *Nodes*; E $\longrightarrow$ *Edges*; k $\leftarrow$ *StretchConstant*;
Input : The graph $G < N, V >$ and $<$ n, d $>\longrightarrow$ set of node-degree pairs;
Description: Attraction force among connected nodes, by updating their (x,y) coordinates to bring them closer to each other.

```
1  begin
2      for i ← 1 to |N| do
3          for j ← 1 to |E_ni| do
4              n1 ← i and n2 ← Other end node of n1
5              Δx ← n1x − n2x
6              Δy ← n1y − n2y
7              Length ← √(Δx × Δx + Δy × Δy)
8              force ← (Length − k) / (k × (100))
9              dx ← force × Δx
10             dy ← force × Δy
11             n1x ← n1x − dx
12             n1y ← n1y − dy
13             n2x ← n2x + dx
14             n2y ← n2y + dy
15         end
16     end
17 end
```

computed based upon the distance between two connected nodes. Our optimized approach is presented in Algorithm 1.

The complexity of attraction force method in force-directed model is $\Theta(E)$, where E represents number of edges in graph. In our power-law based approach its complexity is reduced to $\Theta\left(|V_p|\,|E_p|\right)$, where V_p represents number of Power-Vertices and E_p represents the number of edges of power-vertices in a graph. Moreover, $|V_p| \ll |V|$ and $|E_p| \ll |E|$, i.e the complexity is being reduced.

3.1.3 RepulsionForce Method

This method is based on the force-directed model. The basic principle of this method is to move away the nodes which are not connected by an edge. All non-connected nodes are moved away from each other. A repulsive force is computed based upon the distance between two connected nodes. The optimized method is provided in Algorithm 2.

The major pit fall of force-directed algorithm is the complexity of repulsive force method which is $\Theta(E^2)$, where E represents number of edges in graph. In our power-law based approached, we reduced its complexity

Algorithm 2: RepulsionForce

Data: n $\longrightarrow$ *node*; d $\longrightarrow$ degree;
N $\longrightarrow$ *Nodes*; E $\longrightarrow$ Edges ;
k $\longrightarrow$ Repulsion Constant;
$d_x \longrightarrow$ distance co-efficient of n_1;
$d_y \longrightarrow$ distance co-efficient of n_2;R = Random Value;
$\lambda \longrightarrow$ a constanct initially set to 700;
Input : $<$ n, d $>\longrightarrow$ nodes along their degrees;
Description: Repulsive force between non-connected nodes, by updating their (x,y) coordinates to move them away from each other.

```
1  begin
2      for i ← 1 to N do
3          n1 ← i
4          for j ← i + 1 to N do
5              n2 ← j dx = 0 and dy = 0
6              Δx ← n1 x − n2 x
7              Δy ← n1 y − n2 y
8              Length ← √(Δx × Δx + Δy × Δy)
9              if Length equal to 0 then ;
               // Collision Detection
10
11                 dx = R and dy = R
12             end
13         end
14         else if Length < λ^2 then ;      // Distance
           Limit
15
16             dx ← Δx / Length and dy ← Δy / Length
17         end
18         force ← (n1k × n2k) / 80
19         n1x ← n1x + dx * force
20         n1y ← n1y + dy * force
21         n2x ← n2x − dx * force
22         n2y ← n2y − dy * force
23     end
24 end
```

to $\theta\left(|V_p{}^2|\right)$, where V_p represents of power vertices and $|V_p| \ll |V|$.

3.1.4 UpdateNodesPosition Method

This method changes the node positions once they have been processed by attraction and repulsion forces and updates each node's (x,y) coordinates based on node's temperature and distance co-efficients.

3.1.5 CoolDown Method

This method iteratively reduces the temperature (or heat). As *Temperature* $\propto$ *SpringForces* or we can say at each iteration the heat is being cooled down.

One can easily build such a method through synchronized iterations. We have tuned algorithm temperature cooling mechanism such that while operating on Power-Nodes temperature slowly cools down, as these are core nodes and need maximum temperature through which maximum force value can be exerted upon these nodes. On the other hand, when algorithm calculates forces on Non-Power nodes, temperature falls quickly by rapid cooling as it is least important to exert equal amount of force on non-power nodes as compared to power-nodes.

Logically, power-nodes need high temperature which is slowly decreasing and non-Power should need relatively low temperature which is rapidly decreasing, because of their least importance. By tuning this cooling mechanism, we achieved significant improvement in layout (tightly coupled cluster built around power-nodes) and performance as temperature rapidly decreases on power-nodes thus saving iterations and execution time.

3.2 Time Complexity

The major pitfall in the traditional force-directed algorithm is its complexity which is $\Theta(|V^2| + |E|)$, where V are Vertices and E are Edges. In our algorithm the time complexity for each function is as under:

–Attractive Force $\Rightarrow \Theta(|V_p|\,|E_p|)$
–Repulsive Force $\Rightarrow \Theta(|{V_p}^2|)$
–**Forces Complexity** $\Rightarrow \Theta(|V_p| \cdot (|V_p| + |E_p|))$
  –$V_p \rightarrow$ Number of Power Nodes.
  –$E_p \rightarrow$ Number of Edges connected to Power Nodes
  –Moreover, $V_p \ll V$ and $E_p \ll E$

Force complexity of our proposed power-law based algorithm is significantly reduced as compare to force-directed algorithm's forces complexity. Moreover, the complexity of each function is almost linear therefore total complexity of power-law based algorithm is also linear which indicates significant improvement in complexity.

3.3 Refinements and Optimizations

In the previous section, we claimed that $|V_p| \ll |V|$ and $|E_p| \ll |E|$. We optimized our algorithm along with performance tweaks due to which we are able to gain significant performance as well as more aesthetic and clear visualization of semantic net.

3.3.1 Semantic Based Filtration

It has been observed that in an ontology many concepts are linked with model specific core nodes such as *rdf:Class*. When an ontology is visualized, many edges seem connected to the RDF and OWL specific core nodes. By considering this fact we have applied the filtering process on graph drawing canvas. Figure 5 explains the view-complexity over Amino Acid Ontology of 1,484 Triples which has been reduced due to filtration process.

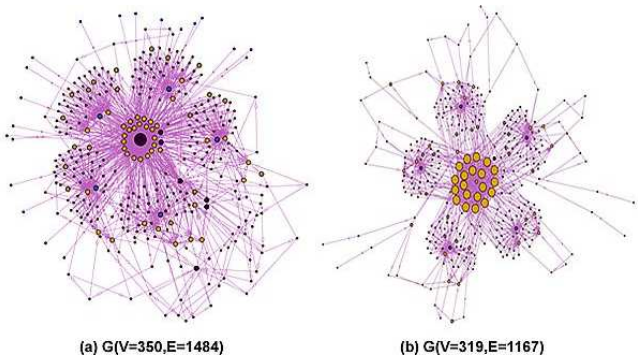


Fig. 5: Semantic filtration (a) Unfiltered graph, (b) Filtered graph

We have filtered primitive constructs of RDF, RDFS, OWL, and XML. Similarly, we also filtered the primitive constructs related to predicates of *RDF, RDFS, OWL, XML*. Any built up edges due to this property are not shown in graph like properties *rdf:Domain, rdf:Range*, however, we do not loss the semantics of visualization by retaining all the information in a tool-tip over nodes. The Table 1 explains the filtered number of nodes and edges as compared to un-filtered.

Table 1: Filtration statistics on nodes and edges

Triples	Unfiltered Graph		Filtered Graph	
	Nodes	Edges	Nodes	Edges
1,515	474	1,515	246	1,245
5,527	3,045	5,527	1,738	3,467
7,330	3,090	7,330	1,052	2,149
10,893	5,937	10,893	3,446	6,830
16,229	8,697	16,629	5,097	10,250
47,003	34,291	47,003	11,767	23,490

4 Implementation Details

We implemented the power-law based semantic net visualization algorithm in a tool called *NavigOwl*[2]. It is developed in Java and for graph drawing we used Piccolo[3]. Jena[4] is used for the processing of semantic models.

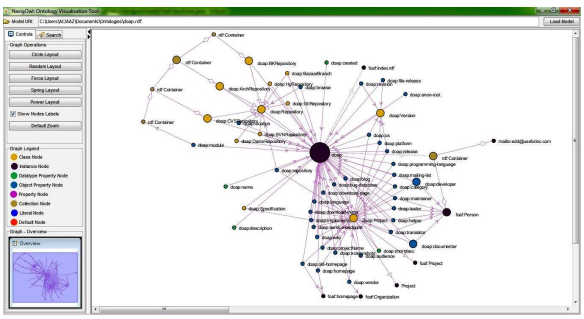


Fig. 6: Snapshot of NavigOwl.

The tool enriched with complete ontology visualization contains whole role-relation hierarchy of each concept (node) and has applied semantic based filtration as we have discussed in Section 3.3.1. This tool supports RDF and OWL ontology files. The snapshot of *NavigOwl* is given in Figure 6. It supports many features as listed below:

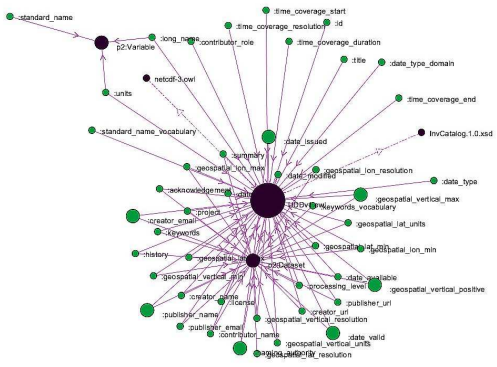


Fig. 7: NavigOwl visualization exhibiting labels of all graph nodes.

–Loads RDF/OWL ontology file and configures its graph by rendering nodes and edges based upon role-relations defined in ontology taxonomy. The node-labels along with edge-relations is shown in visualization in Figure 7.
–Facilitates large-scale semantic nets a.k.a ontologies.
–It recongnizes various pre-configured `RDF/OWL` node types such as *owl:Class* and handles them differently compared to rest of the nodes. This helps in separating the core model specific nodes from the actual ontology concepts.
–Supports fully scalable directed graphs. Visualizes whole role relation hierarchy, defined in ontology. Node's tool-tip exhibits complete role relation hierarchy as shown in Figure 8.
–Zoomable user interface and handling mouse events like pan, drag, mouse-Over, for nodes of a graph.
–Graph overview is also provided to show holistic view of large scale graphs to traverse through whole graph.
–Tool facilitates user to apply various drawing layouts techniques to produce appealing symmetric results of whole graphs.
–Power-law based layout technique produces appealing drawing based upon node-degree distribution, in order to understand node's importance.
–Node search feature is included which highlights the searched node in whole of graph.
–Show / Hide labels of all nodes.
–Node cluttering and edges-overlapping is minimized up to optimum level. However, as in case of large semantic nets, where ontology possess rich role-relation model structure, node-cluttering and edge-overlapping cannot be overcome.

Graph coloring is very important feature used in most of the visualization tools. In case of visualization for large-scale semantic nets, which are enriched with many roles and relationships and ontologies contain different types of concepts, instances and roles among them. We used a color-scheme inspired from Protege to remain consistent. As in an ontology file, different types of relations exist between concepts which are represented by edge line in graph, keeping this concept in view, we have implemented specific arrow shapes and strokes to represent distinct type of edges for user-understandability.

Protege[5] is a famous ontology editor. We have ported NavigOwl in Protege as a tab-widget plugin where users would be able to visualize the RDF / OWL ontology within Protege[6]. Figure 9 shows the snapshot of NavigOwl plugin for Protege.

In Protege there are different view panels like class-hierarchy, object-properties, data-type properties, therefore we have integrated the NavigOwl drawing canvas with Protege class-hierarchy panel. When user selects a particular class node in Protege class-hierarchy panel, that particular node is highlighted on drawing canvas. This functionality helps users to identify selected node.

[2] http://klatif.seecs.nust.edu.pk/navigowl
[3] A 2D graphics API http://www.piccolo2d.org
[4] http://jena.apache.org
[5] http://protege.stanford.edu
[6] http://protegewiki.stanford.edu/wiki/NavigOWL

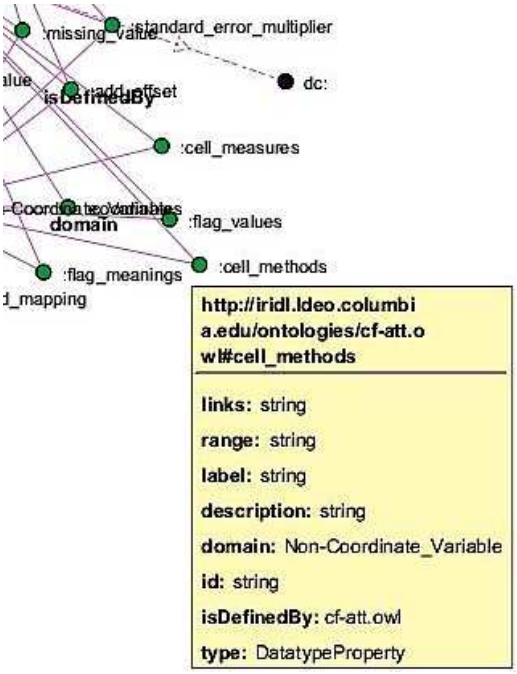


Fig. 8: NavigOwl visualization exhibiting role-relation model of ontology.

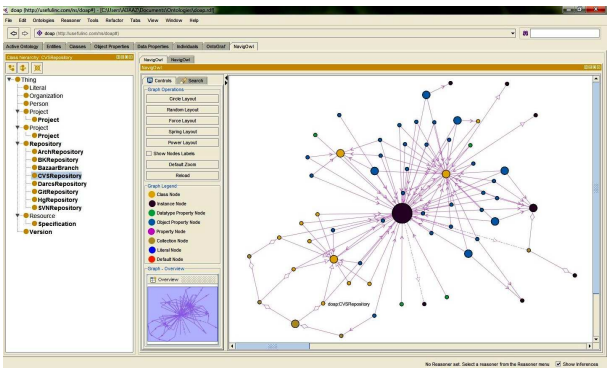
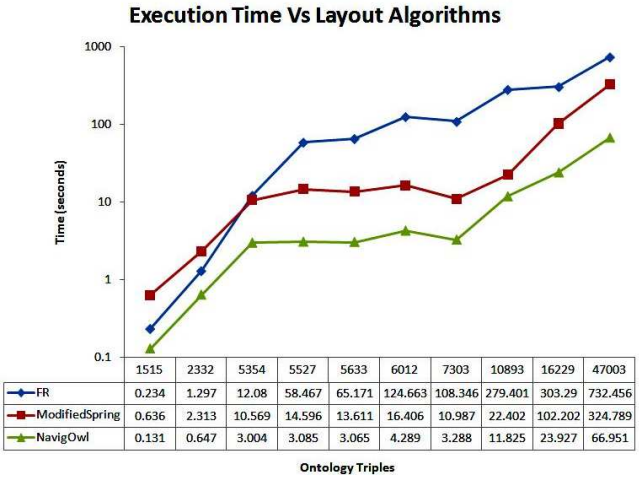


Fig. 9: NavigOwl as plugin of Protege

5 Performance Evaluation

We tested visualization of large scale ontologies on a *Core2 Duo 2.99GHz* machine with 4GB memory. We also implemented two previous approaches of traditional force directed algorithms including *Fruchterman-Reingold* [13], and *modified spring* [21]. We then visualized same ontology on these two algorithms and compared performance results with the proposed algorithm. Figure 10 shows the speedup over existing Force-Directed algorithm approaches.



	1515	2332	5354	5527	5633	6012	7303	10893	16229	47003
FR	0.234	1.297	12.08	58.467	65.171	124.663	108.346	279.401	303.29	732.456
ModifiedSpring	0.636	2.313	10.569	14.596	13.611	16.406	10.987	22.402	102.202	324.789
NavigOwl	0.131	0.647	3.004	3.085	3.065	4.289	3.288	11.825	23.927	66.951

Fig. 10: Comparison of time to layout (in logrithmic scale) of various graph layout algorithms.

5.1 Comparison Over Fruchterman-Reingold

Our implementation showed following observations as compared to the performance impact over Fruchterman-Reingold algorithm:

- –Significant improvement in graph layout by expanding over canvas, symmetrical, minimum edge crossings and almost zero node cluttering as shown in Figure 11.
- –Significant improvement in execution time as shown in Figure 10.
- –Linear improvement in algorithm complexity, as shown in Section 3.2
- –Fruchterman-Reingold algorithm is not highly scalable, takes much longer computation time over large scale ontologies as shown in Figure 10, on 47,003 triples it took 732.456 seconds but our algorithm only took 66.951 seconds.

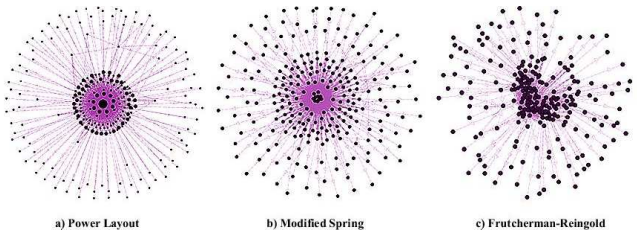


Fig. 11: Layout comparison on OCW Ontology of 1,515 triples filtered graph G(V=246,E=1,245).

5.2 Comparison Over Modified-Spring

Similar implementation showed following observations as compared to the performance impact over Modified spring algorithm [21]:

- –Improvement in graph layout as shown in Figure 11 and 12.
- –Significant improvement in Execution time as in Figure 10.
- –Linear improvement in algorithm Complexity as shown in Algorithm 3.2.
- –Modified-Spring algorithm is not highly scalable, takes much longer computation time over large-scale ontologies.

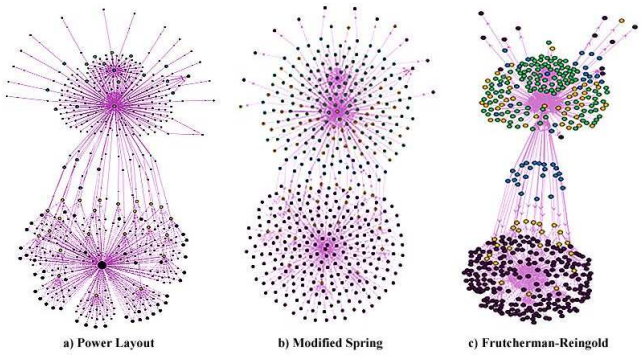


Fig. 12: Layout comparison on Food ontology of 870 triples filtered graph G(V=339,E=604).

5.3 Visualization Results

We have rendered various ontologies in NavigOwl, by applying our power-layout algorithm and obtained promising results in aesthetic preservation of produced graphs with low execution time as compared to force-directed and spring layouts. Table 2 shows these results as follows:

Visualizations of various ontology datasets are shown in Figure 13 and 14

6 Discussion and Case Study

In order to get insight of the complexity of social networks, we have taken *Twitter* as our case study to analyze relationship graph. In order to understand the scheme of relationships and network characteristics, we obtained dataset of *Follower* relationships which contains

Table 2: NavigOwl Results on power-layout

Ontology	**Triples**	$\|V\|$	$\|E\|$	$Time(s)$
GeoNames	104	28	52	0.037
TransOntology Bhakti	195	58	56	0.042
IRI Library CF	378	77	133	0.047
URIplay	597	147	155	0.232
SIOC-NS	615	104	279	0.039
SKOS	1,954	399	1,544	0.146
School	2,178	476	779	0.231
University (LUBH)	5,454	1,095	3,737	2.103
DBPedia	5,633	1,563	1,842	3.198
Barton Subgraph	5,863	1,902	3,691	4.593
Open-BioMed TCM	5,950	2,554	5,098	6.768
TDWG Geography	7,303	1,052	2,149	3.807
LOID OrdnanceSurvey	47,003	11,767	23,490	17.595

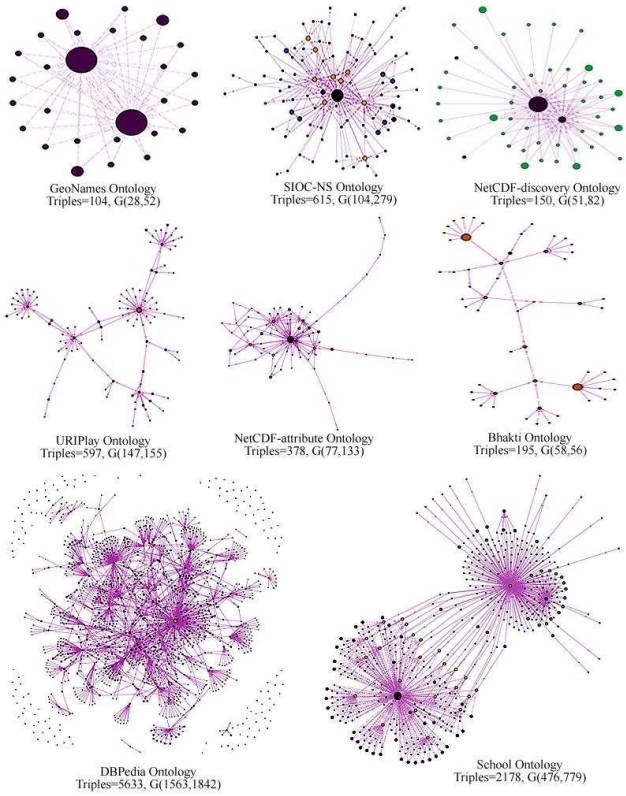


Fig. 13: Symmetrical and clustered graphs of small ontologies.

'who follows who?' type of relationship tuples as shown in Figure 3.

We have created a semantic model of this information transformed all records into that schema in order to visualize it in *NavigOwl* for better understanding of complexity and role-relation hierarchy. The relationship tuples after transformation to semantic model were visualized as shown in Figure 15.

Table 4 represents the mapping of *Twitter* schema to ontology model. We have visualized these networks in *NavigOwl*. After applying power layout we obtained

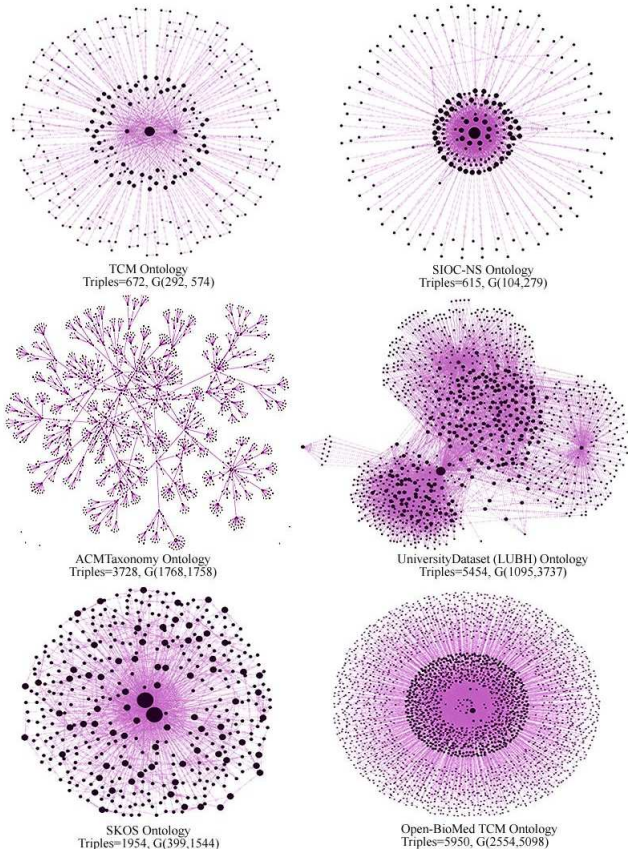


Fig. 14: Large scale symmetrical, dense, clustered visualizations.

Table 3: Tuples represnting *'who follows who?'* in Twitter

Twitter User ID	Twitter Follower ID
6353282	783214
6633812	6353282
7017692	6633812
14951565	7017692
14681199	7017692
8195652	14681199
15015170	8195652
68998614	15015170
3785461	68998614
40887009	3785461
53268444	40887009
—	—

symmetrical, clustered, and aesthetic persistent layouts (c.f. Figure 15).

7 Conclusion and Future Work

This paper proposes the first known solution to mitigate the scalability challenge and performance implications in visualizing large-scale semantic nets. We propose a solution to simultaneously address these open issues by developing modified force-directed layout and exploiting the under-laying concept of power-law degree distribution. The layout was implemented as a visualization tool that can handle large-scale ontologies. We compared our layout algorithm over previous implementations, and achieved significant performance results.

Future direction for this research includes implemention over distributed platform. A distributed layout algorithm can be devised which can be executed over parallel multiple compute nodes and may be computationally more efficient and robust through

Table 4: Mapping of Twitter dataset to ontology schema.

Dataset Records	Ontology Triples	$\|V\|$	$\|E\|$
5,000	532	280	528
10,000	906	473	902
15,000	5,393	2,706	5,389
20,000	11,346	5,663	11,342
30,000	20,533	10,250	20,529
40,000	28,504	14,161	28,500
50,000	36,230	17,929	36,226
60,000	42,649	21,004	42,645

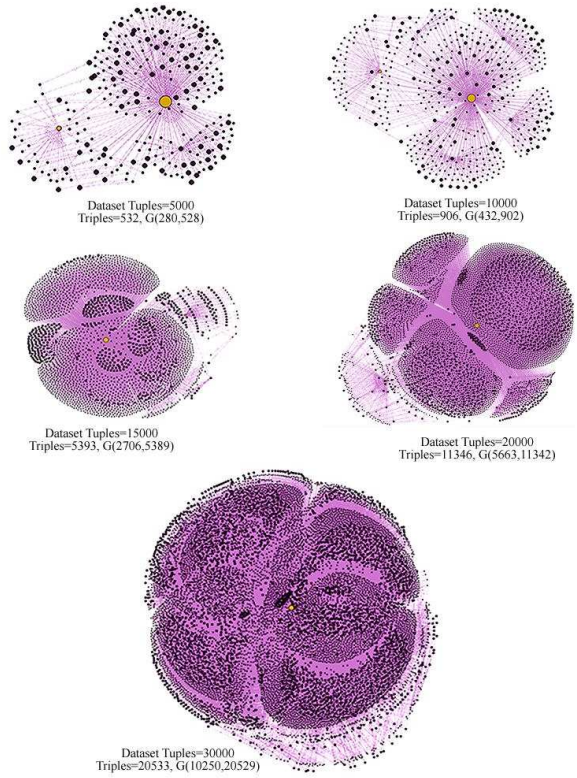


Fig. 15: Twitter Dataset Visualizations on NavigOWL.

exploitation of under-laying parallel computing benefits. Moreover, we also plan to further enrich visualization tool with ontology editing, text searching and SPARQL query options along with formal assessment of our results over more complex large-scale semantic nets.

Acknowledgement

The authors acknowledge the financial support by the ICT R&D Fund (Pakistan) project on Semantic Search and Filtering.

References

[1] H. Akama, M. Miyake, and J. Jung. A New Evaluation Method for the Graph Clustering of Semantic Networks Built from Lexical Co-occurrence Information. *Japan*, 2009.

[2] R. Andersen, F. Chung, and L. Lu. Drawing power law graphs using local/global decomposition. *Twelfth Annual Symposium on Graph Drawing*, 2004.

[3] Mike Bergman. Large-scale rdf graph visualization tools. http://www.mkbergman.com/414/large-scale-rdf-graph-visualization-tools.

[4] R. Bourqui, D. Auber, and P. Mary. How to draw clusteredweighted graphs using a multilevel force-directed graph drawing algorithm. In *Information Visualization, 2007. IV'07. 11th International Conference*, pages 757–764. IEEE, 2007.

[5] F. Boutin, J. Thievre, and M. Hascoet. Focus-based filtering + clustering technique for power-law networks with small world phenomenon. *SPIE-IS & T Electronic Imaging*, 6060, 2006.

[6] D.S.M. Chan, K.S. Chua, C. Leckie, and A. Parhar. Visualisation of power-law network topologies. In *Networks, 2003. ICON2003. The 11th IEEE International Conference on*, pages 69–74. IEEE, 2004.

[7] C. Chen. *Information visualization: Beyond the horizon*. Springer-Verlag New York Inc, 2004.

[8] P Eades. A heuristic for graph drawing. *Congress Numerantium*, 42:149–160, 1984.

[9] P. Eades. Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms and Applications*, 4(3):157–181, 2000.

[10] P. Eades and X. Lin. Spring algorithms and symmetry. *Computing and Combinatorics*, pages 202–211, 1997.

[11] P. Eklund, N. Roberts, and S. Green. Ontorama: Browsing rdf ontologies using a hyperbolic-style browser. In *Cyber Worlds, 2002. Proceedings. First International Symposium on*, pages 405–411. IEEE, 2003.

[12] M. Forman, J. Haralambides, F. T. Leighton Kaufmann, A. Simvonis, E. Welzl, and G. Woeginger. Drawing graphs in the plane with high resolution. *SIAM Journal on Computing*, 22(5):1035–1052, 1993.

[13] T.M.J. Fruchterman and E.M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[14] V. Geroimenko and C. Chen. *Visualizing the semantic web: XML-based internet and information visualization*, chapter Spring-Embedded Graphs for Semantic Visualization, pages 172–182. Springer-Verlag New York Inc, 2006.

[15] M. Girvan and M.E.J. Newman. Community structures in social and biological networks. *PINAS*, 99(12):7821–7826, 2002.

[16] J. Golbeck and P. Mutton. *Force-Directed Drawing Algorithms*, chapter Force-Directed Drawing Algorithms, pages 8493–8957. CRC Press LLC, 2004.

[17] M.L. Huang, P. Eades, and J. Wang. On-line animated visualization of huge graphs using a modified spring algorithm. *Journal of Visual Languages & Computing*, 9(6):623–645, 1998.

[18] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology visualization methods: a survey. *ACM Computing Surveys (CSUR)*, 39(4):10, 2007.

[19] A. Katifori, E. Torou, C. Halatsis, G. Lepouras, and C. Vassilakis. A comparative study of four ontology visualization techniques in prot*égé*: Experiment setup and preliminary results. In *Information Visualization, IV 2006.*, pages 417–423. IEEE, 2006.

[20] M. Lanzenberger, J. Sampson, and M. Rester. Visualization in ontology tools. In *International Conference on Complex, Intelligent and Software Intensive Systems*, pages 705–711. IEEE, 2009.

[21] C.C. Lin and H.C. Yen. A new force-directed graph drawing method based on edge-edge repulsion. *IEEE Computer Society*, 2005.

[22] J. Lin and C. Dyer. *Data-Intensive Text Processing with MapReduce*, chapter Graph Algorithms. University of Maryland, College Park, 2010.

[23] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, 6(2):183–210, 1995.

[24] H. Omote and K. Sugiyama. Method for drawing intersecting clustered graphs and its application to web ontology language. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation-Volume 60*, pages 89–92. Australian Computer Society, Inc., 2006.

[25] H.C. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002.

[26] M. Sarkar and M.H. Brown. Graphical fisheye views of graphs. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 83–91. ACM, 1992.

[27] Wallner.G. Force directed embedding of hierarchical cluster graphs. In *1st International Conference on Relations, Orders and Graphs: Interaction with Computer Science, Mahdia, Tunesia*, volume 17. ROGIS08, 2008.

[28] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Graph Drawing*, pages 31–55. Springer, 2001.

[29] C. Walshaw. A multilevel algorithm for force-directed graph drawing. *Journal of Graph Algorithms and Applications*, 7(3):253–285, 2003.

[30] X. Xu, N. Yuruk, Z. Feng, and T.A.J. Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833. ACM, 2007.

[31] K.P. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated exploration of dynamic graphs with radial layout. In *infovis*, page 43. Published by the IEEE Computer Society, 2001.
[32] N. Yuruk, M. Mete, X. Xu, and T.A.J. Schweiger. A divisive hierarchical structural clustering algorithm for networks. *icdmw*, pages 441–448, 2007.