# Visualisation of Power-Law Network Topologies

David Soen-Mun Chan[1], Khim Shiong Chua[1], Christopher Leckie[1,2] and Ajeet Parhar[3]

ARC Special Research Centre for Ultra-Broadband Information Networks
[1]Department of Electrical and Electronic Engineering
[2]Department of Computer Science and Software Engineering
The University of Melbourne, Parkville 3010, Australia

[3]Telstra Research Laboratories, 770 Blackburn Road, Clayton 3168, Australia
c.leckie@ee.mu.oz.au, ajeet.parhar@telstra.team.com

*Abstract*-- **We present a novel graph layout algorithm called ODL for visualising large network topologies. The main contribution of our algorithm is to simplify the layout problem by separating the nodes in the network into multiple hierarchical layers based on the outdegree of each node. Our algorithm is designed to exploit the underlying structure of power-law topologies, which occur in a wide variety of practical network applications. However, the use of our algorithms is not limited to this class of networks. We have demonstrated that our algorithm can generate useful and aesthetically pleasing layouts for a wide variety of networks, including both regular and power-law topologies. In particular, our algorithm achieved substantial performance improvements over existing layout techniques, including a speed-up factor of up to 260 on real-life Internet routing topologies.**

***Index Terms*—network management, network visualisation, power-law topologies, routing.**

## I. INTRODUCTION

There is a growing need for efficient algorithms to visualise large network topologies in telecommunications. For example, graph layout algorithms can be used in telecommunication network management to help visualise relationships between large numbers of network elements [6], such as the routing relationships between Internet routers [2]. In practice, network managers need to monitor these networks in real-time in order to detect problems due to equipment failures, misconfigurations or unusual traffic patterns. This raises the problem of how to scale-up existing graph layout algorithms in order to visualise large network topologies in real-time. In this paper we present a novel hierachical algorithm, which requires significantly less computation time to layout large networks in comparison to previous approaches [5, 7, 9, 12]. We have implemented this algorithm in a tool that is specifically designed to help network managers analyse large communication networks, such as routing topologies in the Internet.

A promising approach to reducing the complexity of the graph layout problem is to use a hierarchical layout algorithm [12]. In a hierarchical algorithm, the given network is first abstracted by clustering or pruning nodes, and the layout of the abstract network is then optimised. The positions of the nodes in the layout of the abstract network are then used as a starting point to layout the complete network. This approach can be applied recursively to use multiple levels of abstraction. A key issue for hierchical schemes is how to simplify the original network, so that the layout of the abstract network will assist in laying out the entire network. If the abstract network does not reflect the topological structure of the original network, or if the process of abstraction is too complex, then the cost of a hierarchical approach can outweigh its benefits.

We have developed a novel hierarchical approach to graph layout that exploits the *power-law topology* commonly found in many practical networks, such as the Internet [4] and the WWW [1]. A power-law topology has the property that a small proportion of nodes have a high outdegree (i.e., have many connections to other nodes), while the vast majority of nodes have a low outdegree, (i.e., have connections to few nodes). Formally, the frequency $f_d$ of nodes with outdgree $d$ exhibits a power-law relationship of the form $f_d \propto d^a$, $a < 0$. Faloutsos et al. [4] have found that Internet routing topologies follow this power-law relationship with $a \approx -2$.

We use this property to guide the process of hierarchical abstraction in a novel graph layout algorithm. As we move up the abstraction hierarchy, we prune the nodes with the lowest outdegree. Thus, the network at the top of the abstraction hierarchy contains only those nodes with the highest outdegree. We propose that these highly connected nodes are the most influential in structuring the final layout of the complete network, and are thus the best place to start.

In this paper, we present our novel hierarchical algorithm for visualising large network topologies. We demonstrate that this approach to abstraction can produce useful network layouts in significantly less time that other leading layout algorithms on a wide variety of test networks. We begin by summarising previous approaches that have been applied to graph layout. In Section 3, we describe our hierarchical algorithm for visualising power-law network topologies. We then describe our application domain, namely, visualising Internet routing topologies. In Section 5 we demonstrate how our algorithm achieves significantly better performance than existing approaches when tested on real-life Internet routing topologies from the Border Gateway Protocol (BGP).

## II. Previous Approaches to Network Visualisation

Our focus is the layout of large undirected networks with straight-line edges. We are given a network in the form of a graph $G(N,E)$, where $N$ is the set of nodes on the network and $E$ is the set of edges connecting nodes in $N$. Our aim is to find a layout for the network by assigning positions $(x_i, y_i)$ to each node $n_i \in N$. We are interested in algorithms that display the inherent structure of a given network in a succinct fashion. Ideally, we want to maximise the display of symmetry within the network, and preserve the uniformity of edge lengths between pairs of nodes that are identical in topological distance, where the topological distance is the minimum number of hops between nodes. Although it is desirable to minimise edge crossings, we consider this to be of lesser importance because planarity is unlikely to be achievable in the large graphs with high edge density that we have studied.

The most popular types of algorithms for this form of graph layout problem are based on *force-directed* or *spring-embedded* methods [3]. These methods consist of two components: (1) an analogical representation of the graph as a set of electric charges or springs, and (2) an optimisation technique that minimises the energy in the analogical network. The minimum energy state of the analogical network corresponds to an optimal layout of the original graph.

Eades developed the earliest spring-embedded model for graph drawing [3]. He modelled each node as a positive charge and each edge as a spring. The charges (nodes) repelled each other while the springs (edges) attracted the charges. For each node, attractive and repulsive forces are calculated with respect to the other nodes in the network. The positions of the nodes are updated according to the forces on each nodes, until the network settles into a minimum energy state.

A variant of Eades' spring model, the Kamada-Kawai algorithm [7] does not place charges on the nodes. Instead each nodes is connected by springs to every other node, forming a complete graph. The optimum length of each spring reflects the topological distance between each pair of nodes in the original network. Each iteration of the algorithm involves moving a node to a locally optimal position using a gradient descent algorithm. This approach generates aesthetically pleasing results. However, it requires calculations for every pair of nodes in the network, which poses serious complexity issues for large graphs with more than a few hundred nodes.

Another variant by Fruchterman and Reingold [5] simplifies force calculation by using attractive forces only between neighbouring nodes, while repulsive forces are still calculated between all nodes in the graph. Several refinements were introduced, such as only calculating attractive forces between nodes that are in close proximity, and using a *cooling schedule* to limit the maximum movement of nodes at each iteration.

A major drawback of the force-directed approach is the complexity of calculating the forces between all pairs of nodes. An alternative approach is the Inverted Self-Organising Map (ISOM) by Meyer [9]. A form of competitive learning is used to update node positions by using a random input vector or stimulus. The position of the closest node to the stimulus is updated, along with its topological neighbours. By using a restricted update neighbourhood, only a constant number of nodes are updated at each iteration. In addition, Meyer reports experimental results that indicate the number of iterations is linear in the size of the network. However, the resulting visual appearance for complex graphs is often not as symmetric as the results from Kamada-Kawai. This is because the final layout is a side-effect of the local updating procedure, rather than the minimisation of an explicit cost function.

Another alternative is to use a multi-level or hierarchical approach to graph layout. The original graph is abstracted into a hierarchy of simpler graphs, either by clustering similar nodes, or by selecting a small set of influential nodes. A layout is computed for the simplest graph at the top of the hierarchy. This solution is then used as a starting point for the layout of the graph at the next level of detail. The process is repeated at each level of the hierarchy until a layout is found for the original graph. For example, Walshaw [12] used a hierarchical approach that was based on pair-wise clustering of nodes, and force-directed layout. The main advantage of this approach is that a global solution to the layout problem is first made on a simplified graph, which requires less computation time. This solution is then incrementally refined at each subsequent level of detail so that, overall, less time is spent optimising the original graph.

The success of this approach relies on having a suitable strategy for simplifying the graph at each level of abstraction. If the simplification process does not capture the inherent structure of the graph, then the layout of the simplified graph will be of little value when optimising the original graph. In addition, if too many abstraction layers are introduced, then the overhead of processing each level can outweigh the savings. Consequently, we require an efficient strategy for abstracting graphs so that we can easily capture the main topological features that will influence the final layout.

## III. The ODL Layout Algorithm

We have developed a hierarchical layout algorithm called ODL (OutDegree Layout), which is designed to exploit the power-law distribution that is commonly observed for the outdegree of nodes in many kinds of practical networks. Our approach is based on the idea that nodes with a high outdegree play a major role in determining the final structure of the graph, while nodes with a low outdegree have far less effect. Our aims are to decrease the time complexity of graph layout, and generate an aesthetically pleasing layout. We have achieved these aims by using two key features in our algorithm: (1) a hierarchical layering based on node outdegree, and (2) modified layout parameters at each hierarchical layer.

### A. Hierarchical Layering

Our basic approach is to layout the nodes with high outdegree first, then progressively layout nodes with a lower outdegree. Instead of optimising the given network as a whole, nodes with different outdegrees are dealt with separately. We do this by partitioning the graph into different *layers*, where a

layer is a set of nodes with relatively similar outdegree.

Formally, the set of nodes $N$ in the given graph is partitioned into disjoint layers $L_k$, $k = 1...M$. Let $d_i$ denote the outdegree of node $n_i \in N$, i.e., the number of nodes to which $n_i$ is connected. The layers $L_k$ are ordered based on the outdegree of their nodes, where $L_1$ and $L_M$ contain the nodes with the highest and lowest outdegree, respectively.

Our approach is to progressively layout the nodes in each layer $L_k$, given the layout in the preceding layers $L_1...L_{k-1}$. Thus, we begin by laying out the nodes in layer $L_1$. We then fix the position of the nodes in $L_1$, and layout the nodes in layer $L_2$. At the next iteration, we fix the position of the nodes in $L_2$, and layout the nodes in layer $L_3$, and so on.

At each iteration, the layout of the current layer is optimised using a variant of the Fruchterman-Reingold (FR) algorithm [5]. If the complexity of the FR algorithm is $O(|N|2 + |E|)$, then the overall complexity of our algorithm will be $O(\sum|N_k|^2 + \sum|E_k|)$, where $N_k$ and $E_k$ represent the number of nodes and edges in layer $L_k$. For large graphs, this results in a potential saving in comparison to optimising the complete network $O((\sum|N_k|)^2 + \sum|E_k|)$. In practice, we have found that 3 layers are sufficient to achieve a satisfactory layout with improved performance. Our algorithm can be summarised as follows:

Algorithm ODL

1. Sort nodes by outdegree;
2. Partition nodes into layers $L_1...L_M$;
3. For $k = 1…M$

   Layout $L_k$ given node positions in $L_1...L_{k-1}$.

### B. Modified Layout Parameters at Each Hierarchical Layer

In order to minimise the time required for graph layout, we want to limit the movement of nodes in all layers and facilitate the settling of the graph to a minimal energy state as soon as possible. We follow the standard practice of using a *temperature* parameter [5], which represents the maximum distance that a node can be moved in one step. A common approach is to initially use a large temperature setting in order to avoid local minima, and then to decrease the temperature setting using a *cooling schedule* in order to stablise the network as it converges to the final layout.

In our layering approach, we use a different cooling schedule for each layer since the nodes of each of layer have different characteristics. The highest layer possesses a small number of highly influential nodes. Consequently, it requires the highest temperature so that the nodes are forced to spread out from one another, and explore the global search space. This is essential as these highly connected nodes define the final structure of the graph.

In contrast, the lower layers contain a large number of nodes that have only a minor impact on the eventual layout. It is best to limit their freedom of movement so that less time will be spent settling to a minimal energy state. We want to prevent nodes from oscillating around their final positions, as this consumes unnecessary computation time given that the graph is already laid out. Consequently, we use the lowest temperature as well as the quickest cooling schedule at the lowest layer of the hierarchy.

In a similar approach, we can also vary the optimum distance between directly connected nodes at each layer. Nodes at the highest layer need to be well separated so that the final layout is not too cluttered. Similarly, nodes connected at the lowest layer should be separated by a small distance, so that they bunch together to form distinct clusters.

In summary, our algorithm provides a simple and efficient technique for reducing the complexity of graph layout by dividing nodes into layers on the basis of their outdegree. This partitioning is straightforward to compute, and provides a clear basis for fine-tuning the optimisation of the layout at each layer in the hierarchy.

## IV. Visualising Internet routing Topologies

In order to evaluate our hybrid layout algorithm on large networks, we focused on the problem of visualising Internet routing topologies. The role of routing protocols in networks is to ensure that information can be sent between computers connected to the network. Routing protocols are run internally to each of these networks (*intra-domain* routing) as well as between a network and its neighbour (*inter-domain* routing). Inter-domain routing in the Internet is coordinated by the Border Gateway Protocol (BGP) [11]. We have evaluated our layout algorithm using routing topologies generated by BGP as test data. Let us now describe BGP in more detail.

BGP provides a mechanism for advertising reachability between networks that are controlled by different organisations. A group of networks that are controlled by the same organisation is called an Autonomous System (AS). Every AS is given a unique number by the IP addressing authorities. Every ISP has an autonomous system number, as do a number of corporations who employ IP in their private networks. BGP is responsible for inter-domain routing between ASs in the Internet.

Over the past few years, there has been increasing interest in BGP [8], due to the increase in ISPs and the growth in dependence on Internet connections for e-commerce. As the Internet has grown in size and complexity, the interconnection between ASs have been dynamically evolving as ISPs add and eliminate connections to other ASs and companies change ISPs. Furthermore, the contractual agreements between ASs can change due to ISP merging and restructuring. Thus a graph of Autonomous System relationships provides a unique insight into the evolving structure of the Internet.

The nodes of such a BGP network represent the ASs and an edge between vertices denotes that traffic is exchanged between the pair of ASs. The BGP graph (also called an *AS map*) represents logical relationships between ASs. Each AS is represented by a 16-bit number. Not all AS numbers are assigned to administrative domains and some assigned AS numbers are not used. Many ISPs possess several ASs. An AS has its own routers and routing policies and connects to other ASs to exchange traffic with remote hosts.

Regular snapshots of BGP routing information are collected

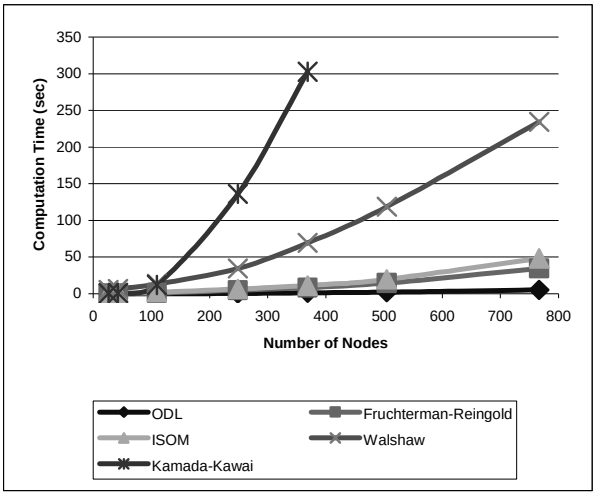


Fig. 1(a). Scalability on AS maps

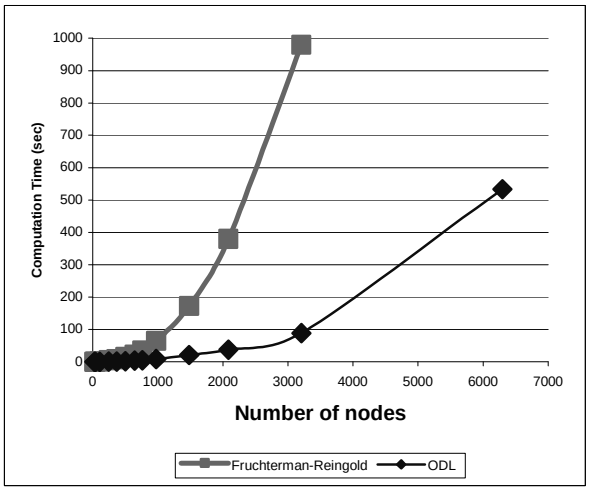


Fig. 1(b). Scalability on large AS maps

by the Route Views Project at the University of Oregon [10]. The Route Views server establishes BGP peering sessions with 53 ISPs in 37 locations. The BGP routing tables are collected once daily from a route server with BGP connections to multiple routers in Japan, United States, Europe and Australia. The largest sample of the BGP dataset has more than 20000 elements and this number is growing steadily with time. A major challenge is how to provide network managers with a meaningful layout of large subsets of this network.

## V. EVALUATION

In order to evaluate the ODL layout algorithm, we compared its performance to several existing layout algorithms discussed in Section 2, namely, Kamada-Kawai (KK) [7], Fruchterman-Reingold (FR) [5], ISOM [9] and Walshaw [12]. We tested these algorithms on two types of networks: (1) Autonomous System (AS) maps derived from BGP routing data as discussed in the previous section, and (2) regular networks that have a clear symmetry or structure. The AS maps enable us to study the effectiveness of our approach on networks with a power-law topology, while the regular networks are used to investigate whether our ODL algorithm is still effective on networks that do not exhibit a power-law topology. We evaluated the five layout algorithms in terms of their *scalability* and *aesthetics*.

Each of the five layout algorithms was implemented in Java, and wherever possible were optimised in order to minimise execution time. In our implementation of ODL, we used 3 layers. Layer $L_1$ contained nodes with outdegree 90 or higher, layer $L_2$ contained nodes with outdegree 21-89, and layer $L_3$ contained nodes with outdegree 1-20. The hardware platform for our evaluation was a Pentium 4 1.4GHz processor with 512MB SDRAM. On this platform we used Java 2 v1.4.1 under the Windows 98 operating system.

### A. Scalability

Our first study was to investigate the scalability of the algorithms, in order to assess their ability to layout large network topologies. Our test data was the BGP routing data described in the previous section. We started with the complete AS map, which represents the connections between all ASs recorded by the Route Views server. We extracted sub-networks of different sizes from the complete AS map, and tested our algorithms on these sub-networks. When we extracted each sub-network, we ensured that the extracted networks still possessed a power-law topology.

The results of the scalability test on BGP data are shown in Figure 1(a). Kamada-Kawai was the least scalable algorithm, rapidly increasing in computation time for networks with more than 400 nodes. The Walshaw algorithm was an improvement, but was still significantly slower than the ISOM, FR and ODL algorithms. We tested the scalability of the FR and ODL algorithms on networks of up to 7,000 nodes (see Figure 1(b)). The ODL algorithm was able to cope best with all of these networks, whereas the FR algorithm was significantly slower beyond 2,000 nodes. At 369 nodes, ODL outperformed the other algorithms by factors of 7.0 (FR), 9.7 (ISOM), 60 (Walshaw) and 265 (KK). At 3,210 nodes, ODL outperformed its nearest rival, FR, by a factor of 11 (see Figure 1(b)). In particular, ODL was significantly faster than the Walshaw algorithm, which was the other hierarchical layout method tested. This is to be expected, since the ODL algorithm uses a much simpler approach to layering, and uses a small number of hierarchical layers.

While the ODL algorithm was motivated by the problem of laying out power-law topology networks, we also wanted to see if ODL is more widely applicable to other types of network topologies. In particular, we wanted to test the effectiveness of ODL on more regular networks that do not follow a power-law topology.

The results are shown in Table 1. The first column gives the name of the test network. The second and third columns give the number of nodes and edges in each network. The remaining columns indicate the CPU time in seconds required

by each layout algorithm on each graph. The bottom row indicates the total CPU time required for all networks.

In terms of total computation time, ODL outperformed the other algorithms by factors of 2.4 (FR), 4.9 (ISOM), 20 (Walshaw) and 86 (KK). ODL is significantly faster than the other layout algorithms even for networks that do not have a power-law topology. Consequently, our heuristic for dividing networks into layers based on the outdegree of nodes has been shown to be effective in a wide range of network topologies.

| Network | \|N\| | \|E\| | ODL | FR | ISOM | Walshaw | KK |
|---|---|---|---|---|---|---|---|
| 13hex | 40 | 52 | 0.06 | 0.10 | 1.04 | 1.78 | 0.66 |
| Ball | 20 | 30 | 0.01 | 0.02 | 0.65 | 1.36 | 0.28 |
| Cuboid | 32 | 64 | 0.03 | 0.06 | 0.93 | 1.68 | 0.27 |
| Grid64 | 72 | 127 | 0.16 | 0.32 | 1.74 | 2.55 | 2.69 |
| Pattern40 | 40 | 77 | 0.05 | 0.12 | 1.09 | 4.40 | 1.21 |
| Star | 19 | 45 | 0.01 | 0.03 | 0.69 | 3.24 | 0.16 |
| Triangle | 253 | 693 | 1.48 | 3.74 | 2.34 | 12.1 | 77.4 |
| Hexwrap | 32 | 44 | 0.05 | 0.07 | 0.96 | 2.54 | 0.05 |
| Heywood | 14 | 21 | 0.02 | 0.01 | 0.56 | 1.54 | 0.06 |
| Hex | 254 | 358 | 1.48 | 3.70 | 6.45 | 35.2 | 204 |
| Total | | | 3.35 | 8.17 | 16.5 | 66.4 | 286.8 |

Table 1. Computation time in seconds of each layout algorithm on regular network topologies

### *B. Aesthetics*

In order to compare the aesthetics of the layouts achieved by each algorithm, we have analysed two different types of networks: a regular cuboid structure containing 32 nodes, and a 369 node subset of the BGP dataset. The results are shown in Figure 2. We have made a qualitative comparison between the algorithms in terms of their ability to (1) display symmetry in the final layout, and (2) find clusters in the data.

The cuboid structure demonstrates the effectiveness of each algorithm in displaying symmetry. KK, FR and ODL all produce layouts that preserve the symmetry of the underlying structure, whereas the Walshaw and ISOM algorithms struggled to generate a regular structure.

An important objective for graph visualisation is to reveal any inherent clusters in the network, i.e., groups of nodes that have a core set of nodes in common. We have used the BGP AS map to test for clusters. The ISOM layout is least successful at identifying clusters. In the Walshaw layout, three clusters easily discernable. It is possible to identify at least four clusters in the KK layout, although there is considerable overlap between the bottom two clusters. There are at least four main clusters revealed by the FR layout, as well as several smaller clusters. Finally, we can identify at least six large clusters from the ODL layout, and at least one smaller cluster.

Overall, the FR and ODL algorithms were the most successful at clustering. The FR algorithm produced the most balanced layout overall. In contrast, the ODL layout was somewhat distorted by the awkward position of the rightmost cluster. Nevertheless, the ODL algorithm generated highly useful layouts with the greatest computational efficiency.

## VI. CONCLUSIONS AND FURTHER WORK

We have presented a novel graph layout algorithm called ODL for visualising large network topologies. The main contribution of our algorithm is that we simplify the layout problem by separating the nodes in the network into multiple hierarchical layers based on their outdegree. Our algorithm is designed to exploit the underlying structure of power-law topologies, which occur in a wide variety of practical network applications. However, the use of our algorithms is not limited to this class of networks. We have demonstrated that our algorithm can generate useful and aesthetically pleasing layouts for a wide variety of networks, including both regular and power-law topologies. In particular, our algorithm achieved substantial performance improvements over existing layout techniques, including a speed-up factor of up to 260 on real-life Internet routing topologies.

Our hierarchical approach enables us to adjust the parameters in the underlying optimisation algorithm to suit each layer in the hierarchy. For further work we are interested in how to fine-tune the optimisation parameters at each layer based on the topological properties of the network.

## ACKNOWLEDGEMENTS

We thank Gerard Wong and Siew Cheong Au for their contributions on visualising BGP networks. This work was supported by the Australian Research Council. The permission of the Chief Technology Officer of Telstra Corporation Limited to publish this paper is hereby acknowledged.

## REFERENCES

[1] R. Albert, H. Jeong and A. Barabasi, "Diameter of the World Wide Web." In *Nature* 401, pp. 130-131, 1999.

[2] H. Burch and B. Cheswick, "Mapping the Internet." In *IEEE Computer*, vol. 32(4), pp. 97-102, April 1999.

[3] P. Eades "A Heuristic for Graph Drawing." In *Congressus Numerantium*, vol.42, pp. 149-160, 1984.

[4] M. Faloutsos, et al., "On Power-Law Relationships of the Internet Topology." In *ACM SIGCOMM*, pp. 251-262, 1999.

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

[6] B. Huffaker, E. Nemeth and K. Claffy. "Otter: A General-Purpose Network Visualization Tool." In *INET'99*, 1999.

[7] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs." In *Information Proc. Letters*, vol 31, pp.7-15, 1989.

[8] C. Labovitz, A. Ahuja, R. Wattenhofer and V. Srinivasan, "The Impact of Internet Policy and Topology on Delayed Routing Convergence." In *IEEE INFOCOM 2001*, pp.537-546, 2001.

[9] B. Meyer, "Self-Organizing Graphs A Neural Network Perspective of Graph Layout." In *Graph Drawing Symposium*, August 1998.

[10] D. Meyer, "Route Views Project Page." http://www.antc.uoregon.edu/route-views/. March 2002.

[11] Y. Rekhter, "A Border Gateway Protocol 4 (BGP-4)." RFC 1771, IETF, March 1995.

[12] C. Walshaw. "A Multilevel Algorithm for Force-Directed Graph Drawing." In *Graph Drawing Symposium*, pp. 171-182, 2000.

***Cuboid (32 nodes & 64 edges)***

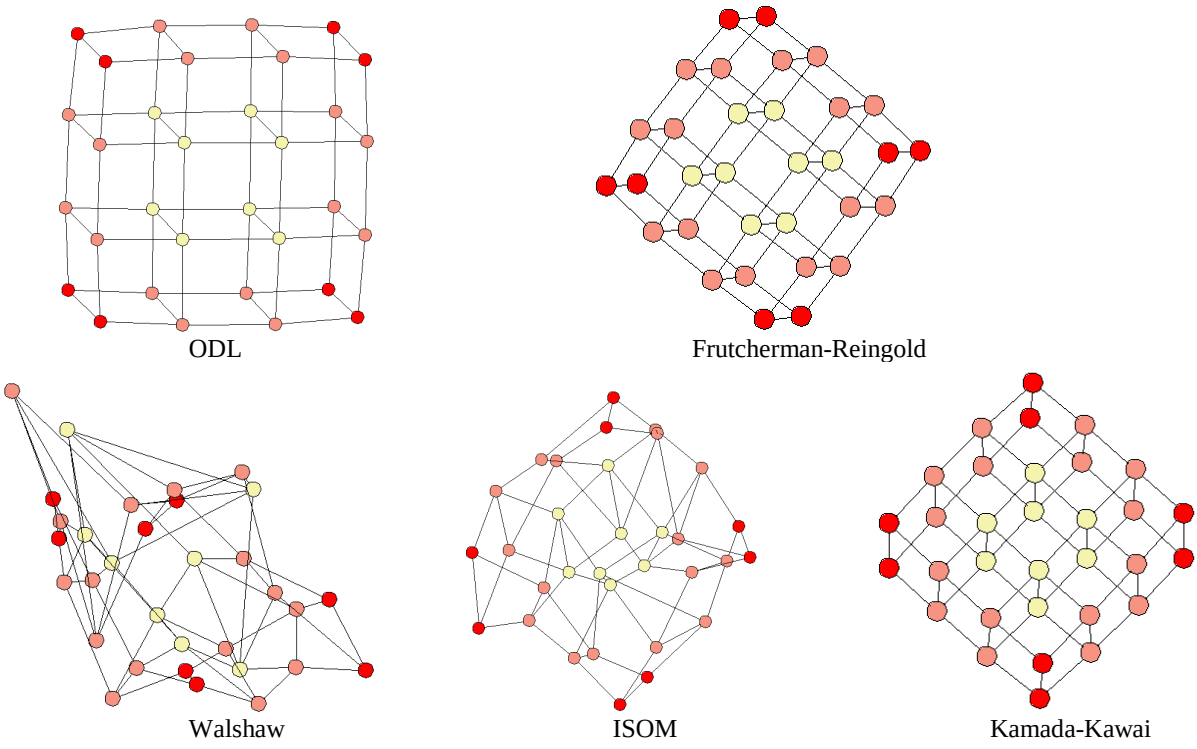


Fig. 2(a). Final layouts of each algorithm for a regular cuboid network

***BGP AS map (369 nodes & 617 edges)***

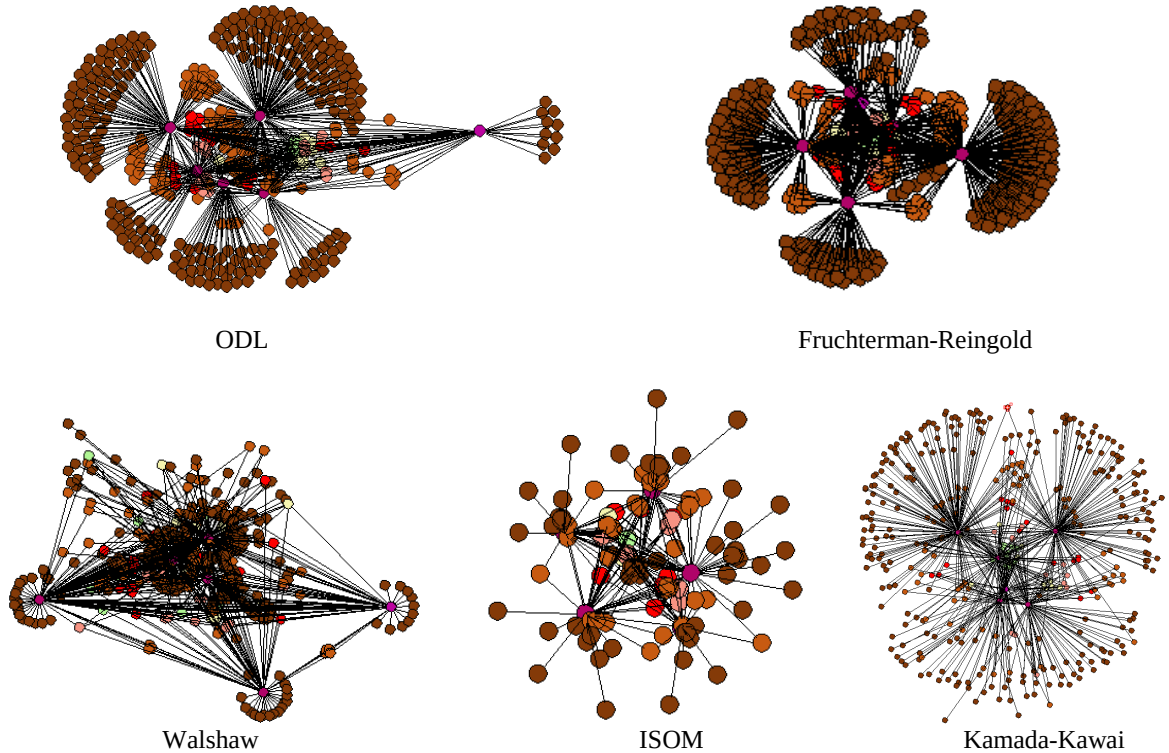


Fig. 2(b). Final layouts of each algorithm for a BGP routing topology network