Force-Transfer: A New Approach to Removing Overlapping Nodes in Graph Layout

Xiaodi Huang and Wei Lai
School of Information Technology, Swinburne University of Technology
PO Box 218. Hawthorn, VIC 3126, Australia
Email: {xhuang, wlai}@it.swin.edu.au

Abstract

Graphs where each node includes an amount of text are often used in applications. A typical example of such graphs is UML diagrams used in CASE tools. To make text information in each node readable in displaying such graphs, it is required there should be no overlapping nodes. This paper proposes the Force-Transfer algorithm to give a new efficient approach to removing overlapping nodes. The proposed approach employs a heuristic method to approximate the global optimal adjustment with the local minimal movement. Scanning from the seed node, the approach orthogonally transfers the minimum forces to only those nodes recursively overlapping with the node from where the forces start. We compare the Force-Transfer with the Force-Scan algorithm by mathematical proofs and experiments. The Force-Transfer approach can generate better results.

Keywords: Graph Layout, Neighbor Nodes, Mental Map, Node Overlapping, Force Transfer.

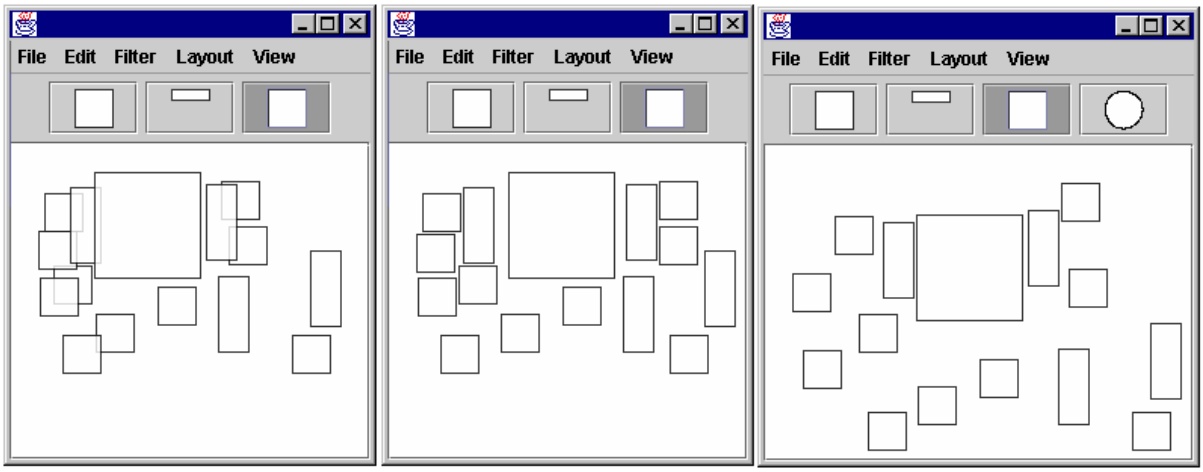
1. Introduction

To remove overlapping nodes in graph layout plays an important role in such a graph where each node includes an amount of text in applications. An example of such a graph is the UML diagram used in CASE tools. These graphs are defined as *practical graphs* [1]. A generic approach for practical graph layout has been proposed [1], that is, to apply a classical graph drawing algorithm to a practical graph first, and then use a post-process to remove overlapping nodes and edge-node intersections [1]. Many classical graph drawing algorithms are available in [3].The approaches for removing overlapping nodes are also potentially applicable to remove overlapping windows in multi-window applications.And they can be applied to layout information for small devices, such as mobile phone, PDA [16].

The approach in [1] for practical graph layout is to make use of existing classical graph drawing algorithms. The critical part of this approach is to develop an efficient post-process for adjusting the graph layout: removing overlapping nodes and edge-node intersections while preserving the mental map [12,14] of the graph layout. The layout adjustment for removing overlapping nodes is also needed after interactive operations, such as enlarge/shrink sub-graph and add /delete nodes, while the metal map of the layout should be preserved.

There are three kinds of approaches to solving the node-overlapping problem: Uniform Scaling [4,7], Force-Based Algorithms [4,14] and Constrained Optimization [2,5,6]. Uniform Scaling is a simple approach to remove overlapping nodes by scaling the graph layout. It preserves the original structure of the graph. However, it may cause nodes to move unnecessarily and the whole graph tends to grow too large. The Force-Based Algorithm for removing overlapping nodes includes two approaches: The Force-Scan Algorithm [4] (FSA) and Cluster Busting in anchored graph drawing [14]. FSA moves the nodes in both horizontal and vertical directions to avoid overlaps. The main idea is to choose a 'force' $\boldsymbol{f}_{vu}$ between two pair of nodes v , u so that if v and u overlap then $\boldsymbol{f}_{vu}$ pushes u away from v. The forces are applied in two scans: one from left to right, one from top to bottom. FSA can preserve the mental map of the original graph and it gives a compact layout compared with Uniform Scaling. Another variation of FSA is to allow the pull force between two nodes; this can make the graph layout as compact as possible. However, this variation of FSA with push and pull forces needs to run FSA a number of times until the forces among nodes become zero [4], so the speed of this approach is slower. The algorithms of Cluster Busting in anchored graph drawing are iterative. The nodes in the graph are iteratively moved according to the measurable criteria. Those criteria are used improve the distribution of the nodes (cluster busting) and minimize the difference between the original layout and adjusted layout (anchored graph drawing). Also those algorithms need to run several iterations to achieve a better-adjusted layout. The Constrained Optimization approach to removing overlapping is based on translating the overlapping node problem into a constrained quadratic optimization one. The optimal objective function is a quadratic expression about differences between initial and adjusted coordinates. This approach is to find an optimal solution to the object function, subject to a collection of constraints [5,6]. This kind of approach can "give better layout than the force scan algorithm, although they are slower" [5]. Other related work includes the SHriMp Algorithm [15], where nodes in the graph uniformly give up screen space to allow a node of interest to grow, and namely the nodes are scaled to fit inside the screen, therefore it is a distortion technique.

This paper introduces a new approach (called Force-Transfer) to removing overlapping nodes, which can give better results (i.e. more compact) than FSA and has the same process speed (i.e. time complexity) as FSA. An example of using our Force-Transfer algorithm (FTA)



(a) An overlapping graph (b) Layout adjusted by FTA (b) Layout adjusted by FSA

Figure 1. An example of overlapping graph adjusted by FTA and FSA

compared to FSA is shown in Figure 1.This demonstrates FTA can remove overlapping nodes and also make the graph layout more compact.

FTA employs a heuristic method to approximate the global optimal adjustment with local minimal movement. The force in this approach is different from that used in FSA. First, we use the minimum force to separate two nodes in the orthogonal direction so it can achieve the local optimisation of adjustment. Second, the force is restrictedly transferred to a dynamic sub-graph that is a cluster of overlapping nodes. In the process of removing overlapping nodes in this sub-graph, some nodes are moved and no longer overlap, but some nodes might newly overlap with nodes outside this cluster sub-graph, so a new sub-graph is formed dynamically. The process will terminate until no such sub-graphs exist. Another feature of FTA is that the first node of the applied force can be any node in the graph. This node is called the *seed node*. Our prototype system can support the user to select a seed node of a graph layout interactively. This can make the whole graph move as little as possible while removing overlapping nodes.

This paper is organized as follows. Section 2 describes FTA in detail. We compare FTA with FSA in section 3. The proofs for the properties of FTA are given in section 4. In section 5, we provide empirical evaluation by using FTA and FSA for a set of arbitrary graph layouts. Finally, we summarize and conclude our work in section 6.

2. The Force-Transfer Algorithm

We assume that a graph G=(V,E,v_s), where V={1,2,…,n} is the set of nodes , $E \subseteq V \times V$ is the set of edges in the graph , and $v_s \in V$, a seed node , which is a special node on which a force will firstly work . Given a node v, its centre coordinate is (x^0_v, y^0_v), and its width and height are w_v, and h_v respectively. Its four corner positions are defined as (x^1_v,y^1_v), (x^2_v,y^1_v), (x^2_v,y^2_v), and (x^1_v , y^2_v) respectively (see node v in Figure 2).

The nodes in G can be sorted by x^1_i (the coordinates of upper left corner (x^1_i,y^1_i) of the node v_i) in horizontal direction, where $i = 1,2, \cdots, |V|$. Similarly, the nodes can be sorted by y^1_i in vertical direction.

2.1 Definitions and Problem Statement

Based on above preliminaries, we give three definitions and formally describe the overlapping problem as follows.

Definition 2.1.1 For each node v, $u \in V$, if they overlap each other, then the following expression should be satisfied:

$$\begin{aligned} &| x^0_v - x^0_u | < (w_v + w_u)/2 + d \quad or \\ &| y^0_v - y^0_u | < (h_v + h_u)/2 + d \end{aligned} \tag{2.1}$$

where d is a minimum gap between nodes v and u. We define them as *Neighbor Node*, denoted by n (u, v), where $v \neq u$. For convenience, we omit d in all the following formulas.

Definition 2.1.2 The *Neighbor Nodes* $NS(v)$ of a node v is a set of the nodes that overlap with node v,

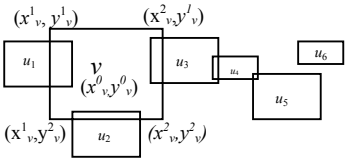


Figure 2. *Neighbor Nodes*

that is, $NS(v) = \{u \mid n(v,u), u \in V\}$. In Figure 2, $NS(v) = \{u_1, u_2, u_3\}$. We also give the definitions of *Left, Right, Up and Down Neighbor Nodes* of a node v as follows:

$$LNS(v) = \{u \mid x^1_u < x^1_v, u \in NS(v)\}$$
$$RNS(v) = \{u \mid x^1_u \geq x^1_v, u \in NS(v)\}$$
$$UNS(v) = \{u \mid y^1_u < y^1_v, u \in NS(v)\}$$
$$DNS(v) = \{u \mid y^1_u \geq y^1_v, u \in NS(v)\}$$

Obviously, in Figure 2, we have $LNS(v)=\{u_1\}$, $RNS(v)=\{u_2, u_3\}$, $UNS(v) = \emptyset$, and $DNS(v) = \{u_1, u_2, u_3\}$.

Definition 2.1.3 The *Transfer Neighbor Nodes* of a node *v*, *TNS* (*v*), includes all the nodes around node *v* in which every node at least overlaps with one other node. The procedure of defining *TNS* (*v*) is as follows: find *Neighbor Nodes* of a node *v*, *NS* (*v*), and then all the nodes in *NS*(*v*) in turn find their *Neighbor Nodes* and so on, until the node that has no *Neighbor Nodes* is reached. So the recursive definitions of *TNS* (*v*) maybe be given like this:

i=1: $TNS(v)_1 = \left(\bigcup_{u_j \in NS(v)} NS(u_j) \right) \bigcup NS(v) - \{v\}$

i>1:

$$TNS(v)_i = \left(\bigcup_{u_j \in TNS(v)_{i-1}} NS(u_j) \right) \bigcup TNS(v)_{i-1} - \{v\}$$

then

$$TNS(v) = TNS(v)_i$$

where $i = 1,2,\ldots,|V-1|$, and until the expression, $TNS(v)_i = TNS(v)_{i-1}$, is always satisfied.

In Figure 2, according to above definitions, for example, we have:

$$\begin{aligned} TNS(v)_1 &= NS(u_1) \cup NS(u_2) \cup NS(u_3) \\ &\quad \cup \{u_1, u_2, u_3\} - \{v\} \\ &= \{u_1, u_2, u_3, u_4\} \\ TNS(v)_2 &= NS(u_1) \cup NS(u_2) \cup NS(u_3) \cup NS(u_4) \\ &\quad \cup \{u_1, u_2, u_3, u_4\} - \{v\} \\ &= \{u_1, u_2, u_3, u_4, u_5\} \end{aligned}$$

therefore, we have $TNS(v) = \{u_1, u_2, u_3, u_4, u_5\}$.

Similarly, we can define the *Left, Right, Up and Down Transfer Neighbor Nodes*, i.e. $LTNS(v) = \{u_1\}$, $RTNS(v) = \{u_2, u_3, u_4, u_5\}$, $UTNS(v) = \emptyset$, $DTNS(v) = \{u_1, u_2, u_3, u_4, u_5\}$.

From above definitions, obviously, we have:

Corollary (1) $TNS(v) \supseteq NS(v)$; (2) $TNS(v) = RTNS(v) \cup LTNS(v)$; (3) $TNS(v) = UTNS(v) \cup DTNS(v)$; (4) $RTNS(v) \cap LTNS(v) = \emptyset$; (5) $UTNS(v) \cap DTNS(v) = \emptyset$.

Now, we can formalize the problem of removing overlapping nodes. If the following expression

$$NS(v_i) = \emptyset \quad \text{, for } \forall v_i \in V$$

where $i = 1,2 \cdots, |V|$, is satisfied, and then the graph has no overlapping nodes.

For every node $v, u \in V$, if the nodes v, u are separated, the following expression should be satisfied:

$$|x_v^0 - x_u^0| \geq (w_v + w_u)/2 \quad \text{or}$$

$$|y_v^0 - y_u^0| \geq (h_v + h_u)/2$$

In Figure 3(a), for example, if a node v is kept still, in order to remove the overlapping node u, then the direction of the applied force on node u can be from anywhere , but its orthogonal projection must make node u move to right/left or to down/up along orthogonal direction at a distance of either Δx_{vu} or Δy_{vu}. FTA calculates the overlapping distance between nodes v, u in both horizontal and vertical directions:

$$\begin{aligned} \Delta x_{vu} &= \min_{n(v,u);v,u \in V} \{|x_v^2 - x_u^1|, |x_u^2 - x_v^1|\} \\ &= (w_v + w_u)/2 - |x_v^0 - x_u^0| \end{aligned} \tag{2.2}$$

$$\begin{aligned} \Delta y_{vu} &= \min_{n(v,u);v,u \in V} \{|y_v^2 - y_u^1|, |y_u^2 - y_v^1|\} \\ &= (h_v + h_u)/2 - |y_v^0 - y_u^0| \end{aligned} \tag{2.3}$$

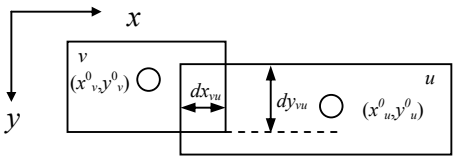
then chooses the minimum magnitude of Δx_{vu} and Δy_{vu} as a "force" to work on u to push u away from v:

$$|\boldsymbol{f}_{vu}| = \min_{\boldsymbol{n(v,u);v,u \in V}} \{\Delta x_{\boldsymbol{vu}}, \Delta y_{\boldsymbol{vu}}\}$$

so we have:

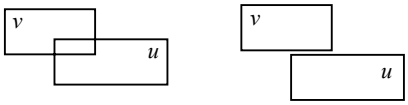
$$\boldsymbol{f}_{vu} = \begin{cases} \Delta x_{vu}\boldsymbol{i} & if \ \Delta x_{vu} \leq \Delta y_{vu} \wedge x_v^1 \geq x_s^1 \wedge x_u^1 \geq x_s^1 \\ -\Delta x_{vu}\boldsymbol{i} & if \ \Delta x_{vu} \leq \Delta y_{vu} \wedge x_v^1 < x_s^1 \wedge x_u^1 < x_s^1 \\ \Delta y_{vu}\boldsymbol{j} & if \ \Delta x_{vu} > \Delta y_{vu} \wedge y_v^1 \geq y_s^1 \wedge y_u^1 \geq y_s^1 \\ -\Delta y_{vu}\boldsymbol{j} & if \ \Delta x_{vu} > \Delta y_{vu} \wedge y_v^1 < y_s^1 \wedge y_u^1 < y_s^1 \\ 0 & otherwise \end{cases} \tag{2.4}$$

where $\boldsymbol{i}$, $\boldsymbol{j}$ is a unit vector in the x, y direction respectively. The scan starts from the seed node v_s to those nodes that are located at its right, left, up and down directions, respectively. For example, In Figure 3 (b), $|\boldsymbol{f}_{vu}^x| = \Delta x_{vu}, and \ |\boldsymbol{f}_{vu}^y| = 0$, due to $\Delta x_{vu} < \Delta y_{vu}$, while in Figure 3 (d) $|\boldsymbol{f}_{vu}^y| = \Delta y_{vu}, and \ |\boldsymbol{f}_{vu}^x| = 0$ due to $\Delta x_{vu} > \Delta y_{vu}$.



(a) Overlap distances

(b) Overlap graph 1 (c) Non-overlap graph 1



(d) Overlap graph 2 (e) Non-overlap graph 2

Figure 3. Overlap calculation

Figure 3 (c) and Figure 3 (e) are the results of applying $\boldsymbol{f}_{vu}^x$ and $\boldsymbol{f}_{vu}^y$ to Figure 3 (b) and Figure 3 (d) respectively.

2.2 Force-Transfer Model

Our approach to adjusting a graph layout is to choose a seed node first. Then there are four scans starting from the seed node in right, left, down, and up directions. The Force-Transfer model is applied to the process of each scan. In this section, we use the scan in the right direction to explain the Force-Transfer model.

We define a cluster sub-graph as: $C(v) = TNS(v) \cup \{v\}$, which is a cluster of overlapping nodes (see Figure 4). To get the boundary of a cluster sub-graph, we use a rectangle to include this sub-graph. We consider this rectangle as a new cluster node (i.e. a dummy node) denoted by v_C. In Figure 4, we use a dash-line box to represent this dummy node. Its four corner positions are $(x^1_{v_C}, y^1_{v_C})$, $(x^2_{v_C}, y^1_{v_C})$, $(x^2_{v_C}, y^2_{v_C})$, and $(x^1_{v_C}, y^2_{v_C})$ clockwise. The coordinates of $x^1_{v_C}$, $y^1_{v_C}$, $x^2_{v_C}$ and $y^2_{v_C}$ can be calculated as follows:

$$x^1_{v_C} = \min_{v_i \in C(v)} \{x^1_{v_i}\},\ y^1_{v_C} = \min_{v_i \in C(v)} \{y^1_{v_i}\}$$

$$x^2_{v_C} = \max_{v_i \in C(v)} \{x^2_{v_i}\},\ y^2_{v_C} = \max_{v_i \in C(v)} \{y^2_{v_i}\}$$

The graph G can be accordingly regarded as an union of C(v), C(u), …, etc. (see Figure 4). Suppose the seed node v_s is included in C(v). We assume the scan begins with the seed node and consider that the scan direction is from left- to- right (see Figure 4).

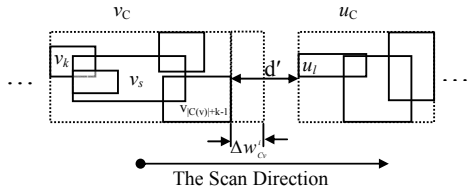


Figure 4. Cluster nodes and Force Transfer

In the process of this scan, the forces are applied to the nodes in $RTNS(v_s)$, that is, the force $\boldsymbol{f}_{ij}$ between the nodes v_i, v_j in $RTNS(v_s)$ will also be transferred to the nodes in $RTNS(v_j)$. So, for the node v_i in C(v), where $s < i \le |C(v)| + k - 1$ and k is the first subscript of all ordered nodes in C(v), the total forces on the node v_i are as follows:

$$\boldsymbol{F}^x_i = \sum_{j=s}^{i-1} \boldsymbol{f}^x_{j(j+1)}$$

where $\boldsymbol{f}^x_{j(j+1)}$ are calculated by formula (2.4). Since the positions of the overlapping nodes will be moved after the beginning of the scan, the width of the cluster node v_C maybe be expanded in right direction. After scanning the node v_i, the width of v_C has been expanded by Δw^i_{Cv}, and obviously we have:

$$\Delta w^i_{Cv} = |\boldsymbol{F}^x_i| = |\sum_{j=s}^{i-1} \boldsymbol{f}^x_{j(j+1)}|$$

so if there exists an integer k_0, $s \le k_0 \le i-1$, and the following expression is satisfied:

$$\Delta w^i_{Cv} \ge d' = dist(v_C, u_C) - (w_{v_C} + w_{u_C})/2 \quad (2.5)$$

where $dist(v_C, u_C)$ is the distance between the central positions of the cluster nodes v_C and u_C, and w_{v_C}, w_{u_C} is their original widths respectively , then one of the cluster nodes v_C , u_C becomes a *Neighbor Node*. Therefore, the applied forces between the nodes in $RTNS(v_{k_0-1})$,i.e. from the node v_{k_0} to $v_{|C(v)|+k-1}$ in C(v), will also be transferred to all the nodes in C(u). Namely, a new cluster node is dynamically created, which is composed of both the nodes in $RTNS(v_{k_0-1})$ and those in C(u).

We conclude that the transfer function, which describes totally applied forces on the node v_i in C(v), or C(u), is as follows:

$$\boldsymbol{F}^x_i = \begin{cases} \sum_{j=s}^{i-1} \boldsymbol{f}^x_{j(j+1)} & s \le i \le |C(v)| + k - 1 \\ \sum_{j=k_0}^{i-1} \boldsymbol{f}^x_{j(j+1)} & l \le i \le |C(u)| + l - 1 \ \wedge \ if(2.5)\ is\ satisfied \\ \sum_{j=l}^{i-1} \boldsymbol{f}^x_{j(j+1)} & l \le i \le |C(u)| + l - 1 \ \wedge \ if(2.5)\ is\ not\ satisfied \end{cases} \quad (2.6)$$

where $s \le k_0 \le i-1$, k_0 is an integer and l is the first subscript of all ordered nodes in C(u).

We can give similar transfer functions of the left, up and down scan directions.

In FTA, the transfer function is used to dynamically find the *Transfer Neighbor Nodes.*

2.3 Pseudo-Code of the Force-Transfer Algorithm

In the following pseudo-code, we assume that given G = (V,E,v_s), and x^1_i are the upper left corner x coordinates of the nodes, where $i = 1,2,\cdots,|V|$, and v_s is the seed node. The Horizontal Transfer works in two directions: from the seed node to the right nodes, and from the seed node to the left nodes.

Horizontal Transfer:

Assume that the nodes have been sorted by their x^1_i coordinates
$i \leftarrow s$
while $i <$ |V| do{Right Horizontal Scan}
  Find $RNS(v_i)$
  If $RNS(v_i) \neq \emptyset$ then
    Find $RTNS((v_i)$
    $j = \arg\min_j \{v_j \mid v_j \in RNS(v_i)\}$
    $|f^x_{ij}| := x^2_i - x^1_j$
    $|f^y_{ij}| := \min\{|y^2_i - y^1_j|, |y^1_i - y^2_j|\}$
    $\delta \leftarrow \min\{|f^x_{ij}|, |f^y_{ij}|\}$
    If $\delta = |f^x_{ij}|$ then

// Right Horizontal Transfer
for each $v_j \in RTNS(v_i)$ do

$$x^0_{v_j} := x^0_{v_j} + \delta + d$$

end{for}
end if
end if
$i \leftarrow i+1$
end{Right Horizontal Scan}

$i \leftarrow s$
while i <0 {Left Horizontal Scan}
If Find $LNS(v_i) \neq \emptyset$ then
Find $LTNS((v_i)$
$j = \arg\max_j \{v_j \mid v_j \in LNS(v_i)\}$
$|f^x_{ij}| := x^1{}_i - x^2_j$
$|f^y_{ij}| := \min\{|y^1{}_i - y^2_j|, |y^2{}_i - y^1_j|\}$
$\delta \leftarrow \min\{|f^x_{ij}|, |f^y_{ij}|\}$
If $\delta = |f^x_{ij}|$ then
//Left Horizontal Transfer
for each $v_j \in LTNS(v_i)$ do

$$x^0_{v_j} := x^0_{v_j} - \delta - d$$

end{for}
end if
end if
$i \leftarrow i+1$
end{Left Horizontal Scan}

A vertical scan is similar. FTA includes horizontal and vertical scans. That is, the force transfer starts from the seed node and proceeds to other nodes in the four orthogonal directions.

2.4 Choice of the Seed Node

From previous description, we know that the seed node plays an important role in our approach. In order to choose an appropriate node as the seed node, we define the cost function as the sum of the moved distances of all the nodes in V after layout adjustment:

$$f_{cost}(v_s) = \sum_{v \in V} (\Delta x_v + \Delta y_v)$$

The candidate of the seed node should enable to minimize the value of above function. Thus, the problem can be translated into an optimization problem, that is, to minimize the objective function $f_{cost}(v_s)$, subject to the condition of removing all the overlapping nodes. Many methods can be used to find the solution, such as Simulated Annealing [8], Genetic algorithm [9] and Tabu Search [10, 11], etc. In order to approximate the global optimization, those algorithms try to escape from getting trapped in local minima by different ways. Applying those approaches to our problem, we maybe find an optimal point as the centre of the seed node. If the point is not the centre of a node in V, we can regard it as a dummy node. However, the process of finding the optimal point is time-consuming. We therefore have to employ a heuristic method to approximate the optimum. Our experiments show that: in most cases, choosing the leftmost node as the seed node can give a better layout, but for the symmetric overlapping graph, the centre node of the graph should be chosen. Another choice is to let the user select the seed node for a fast graph layout.

3. Comparison With the Force-Scan Algorithm

FTA is based on FSA, but there are some differences as follows.

Applied forces: FTA chooses only one of the minimum magnitude projections of the forces to separate two overlapping nodes. So one of the overlapping nodes is only moved in the horizontal or vertical direction. However in FSA the node is moved in both orthogonal directions because the forces have two projections. The forces used in FTA and FSA are shown in Figure 5.

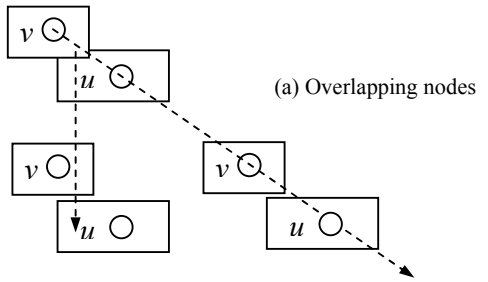


Figure 5. The forces in FSA and FTA

Transfer of the forces: In the right scan of FTA, for example, the forces are only transferred to the nodes in a dynamic sub-graph (a cluster of overlapping nodes) on the right, i.e., to the nodes in the *Transfer Neighbor Nodes* of the node where the applied forces start. The forces in FSA are transferred to all the right nodes in the graph from the nodes of the applied forces.

Seed node: In FTA, a seed node can be any node in the graph, even a dummy node, while in FSA it is only the leftmost one.

Position of the node: In FSA, the centre coordinates of the nodes are used for the force calculation. In FTA, the top-left corner coordinates of the node are used.

Compared with FSA, FTA gives a nicer and compact layout.

4. Properties of the Force-Transfer Algorithm

In this section, we discuss some properties of FTA.

Theorem 1 The output of FTA is a more compact graph layout than the output of FSA.

Proof. We use the cost function previously defined to show above result. The smaller the value of the cost function is, the more compact the layout is, because the

graph layout has less been changed. That is, we need to prove: $f_{cost}^{FTA}(v_s) \le f_{cost}^{FSA}(v_s)$.

In FTA, from (2.2), (2.3) and (2.4), we have:

$$\delta_{ij}^x = |\boldsymbol{f}_{ij}^x| = \begin{cases} (w_i + w_j)/2 - |x_i^0 - x_j^0| & if \ \Delta x_{ij} \le \Delta y_{ij} \ \wedge \ n(i,j) \\ 0 & otherwise \end{cases}$$

and

$$\delta_{ij}^y = |\boldsymbol{f}_{ij}^y| = \begin{cases} (h_i + h_j)/2 - |y_i^0 - y_j^0| & if \ \Delta x_{ij} > \Delta y_{ij} \wedge n(i,j) \\ 0 & otherwise \end{cases} \quad (2.10)$$

Similarly, In FSA, based on [7,13], we have:

$$\delta_{ij}^x = \begin{cases} (w_i + w_j)/2 - |x_i^0 - x_j^0| & if \ \Delta x_{ij} \le \Delta y_{ij} \wedge n(i,j) \\ \phi_{ij}^x & if \ \Delta x_{ij} > \Delta y_{ij} \wedge n(i,j) \wedge y_i^0 \ne y_j^0 \\ 0 & otherwise \end{cases}$$

$$\delta_{ij}^y = \begin{cases} (h_i + h_j)/2 - |y_i^0 - y_j^0| & if \ \Delta x_{ij} > \Delta y_{ij} \wedge n(i,j) \\ \varphi_{ij}^y & if \ \Delta x_{ij} \le \Delta y_{ij} \wedge n(i,j) \wedge x_i^0 \ne x_j^0 \\ 0 & otherwise \end{cases} \quad (2.11)$$

where

$$\varphi_{ij}^x = (1 + \frac{h_i + h_j}{2(y_i^0 - y_j^0)})(x_i^0 - x_j^0)$$

and

$$\varphi_{ij}^y = (1 + \frac{w_i + w_j}{2(x_i^0 - x_j^0)})(y_i^0 - y_j^0)$$

We suppose the seed node $v_s = v_1$, the leftmost node in V. For FTA, in the worst case, like FSA, all the applied forces between node v_i and node v_j will be transferred to all the nodes which are located on the right side of node v_j, so that in both algorithms, the total adjustment distances are

$$f_{cost}(v_1) = \sum_{i=2}^{|V|} (\sum_{j=1}^{i-1} \delta_{ij}^x + \sum_{j=1}^{i-1} \delta_{ij}^y) \quad (2.12)$$

We also suppose there are M_i pairs of overlapping nodes between nodes v_i ,v_j, and among them there are m_j pairs which meet $\Delta x_{ij} \le \Delta y_{ij}$, where $0 \le m_i \le M_i \le j - i$, and M_i and m_i are integers.

For the right scan of node v_i to adjust its overlapping nodes, in FTA, all nodes in its *Transfer Right Neighbor Nodes*, *TRNS*(v_i), will be moved to right at a distance of δ_{ij}^x if the overlapping distance $\Delta x_{ij} \le \Delta y_{ij}$, so the totally moved distances are $\delta_{ij}^x \ |TRNS(v_i)|$. According to (2.10) and (2.12), we have:

$$f_{cost}^{FTA}(v_1) = \sum_{i=1}^{|V|} \{m_i \ |TRNS(v_i)| \ \delta_{ij}^x + (|TRNS(v_i)| - m_i)\delta_{ij}^y\} \quad (2.13)$$

However, in FSA, all the nodes on its right (|V|-*i* nodes) will be moved at distances of both δ_{ij}^x (or δ_{ij}^y) and ϕ_{ij}^y (or ϕ_{ij}^x). Therefore, according to (2.11) and (2.12), we have:

$$f_{cost}^{FSA}(v_1) = \sum_{i=1}^{|V|} \{m_i(|V| - i)\delta_{ij}^x + m_i(|V| - i)\varphi_{ij}^y + (|TRNS(v_i)| - m_i)(|V| - i)\delta_{ij}^y + (|TRNS(v_i)| - m_i)(|V| - i)\varphi_{ij}^x\} \quad (2.14)$$

in both (2.13) and (2.14), where

$$j = \arg\min_j \{v_j \mid v_j \in RNS(v_i)\}$$

Obviously, $|TRNS(v_i)| \le |V| - i$, and from (2.13) and (2.14), we have $f_{cost}^{FTA}(v_1) \le f_{cost}^{FSA}(v_1)$. If $v_s \ne v_1$, According to the definition of the cost function and the node v_s is always the node v_1 in FSA, then we can deduce $f_{cost}^{FTA}(v_s) \le f_{cost}^{FTA}(v_1) \le f_{cost}^{FSA}(v_1) = f_{cost}^{FSA}(v_s)$. Intuitively, the reason is that FTA moves the nodes only by a distance of δ_{ij}^x or δ_{ij}^y in horizontal or vertical direction for every two overlapping nodes, whereas FSA by distances of both δ_{ij}^x(or δ_{ij}^y) and ϕ_{ij}^y(or ϕ_{ij}^x) in both horizontal and vertical directions.

Additionally, in most cases, $f_{cost}^{FTA}(v_s) < f_{cost}^{FSA}(v_1)$; only in special cases, such as two overlapping nodes and their x or y centre coordinates are equivalent, then $f_{cost}^{FTA}(v_s) = f_{cost}^{FSA}(v_1)$.

Theorem 2 FTA ensures its output has no overlapping nodes.

Proof. For any two nodes v_i and u_j on the same side of the seed node v_s, we need to prove $NS(v_i) = \phi$ or $NS(u_j) = \phi$. Suppose their original and new *x* coordinates are $x_i^0, x_j^0, x_i^{0'}, x_j^{0'}$, respectively, and $x_j^0 \ge x_i^0$, so we have

$$\begin{aligned} x_j^{0'} - x_i^{0'} &= x_j^0 + \sum_{n=s+1}^{j} \sum_{m=s}^{n} \delta_{mn}^x - (x_i^0 + \sum_{n=s+1}^{i} \sum_{m=s}^{n} \delta_{mn}^x) \\ &= x_j^0 - x_i^0 + \sum_{n=s+1}^{j-i} \sum_{m=s}^{n} \delta_{mn}^x \\ &\ge x_j^0 - x_i^0 + \delta_{mn}^x \\ &= x_j^0 - x_i^0 - |x_i^0 - x_j^0| + \frac{w_i + w_j}{2} \\ &= \frac{w_i + w_j}{2} \end{aligned}$$

Similarly, for $y_j^0 \ge y_i^0$, thus

$$y_j^{0'} - y_i^{0'} \ge \frac{h_i + h_j}{2}$$

That is, the forces applied to these two nodes can separate them either in x or y direction with a distance of either $x_j^{(t)} - x_i^{(t)} \geq \frac{w_i + w_j}{2}$ or $y_j^{(t)} - y_i^{(t)} \geq \frac{h_i + h_j}{2}$. If the nodes v_i , u_j are on different sides of the seed node v_s, for example, on left and right sides, we also can give a similar proof. Therefore we have $NS(v_i) = \phi$ or $NS(u_j) = \phi$.

Theorem 3 The time complexity of FTA is $O(n^2)$.

FTA cannot guarantee to preserve the "orthogonal ordering" mental map model [12]. Since the forces are only transferred between the neighbor nodes, the relative positions between them will be kept. However, it cannot guarantee to keep the relative positions between neighbored nodes and non-neighbored nodes. So, we need to change FTA a little to preserve the "orthogonal ordering" model. That is, in the process of the force-transfer for each node, we move all the successive nodes in the scan directions, no matter whether they are neighbor nodes or not. This variation of FTA will produce a less compact graph layout, but it preserves "orthogonal ordering".

5. Empirical Evaluation

We have implemented FTA and FSA in a prototype called PGD by using the Java programming language (see Figure 6). In this section, we compare a set of arbitrary graph layout, some of which are similar to those in [5], by applying FTA and FSA, and discuss some features.

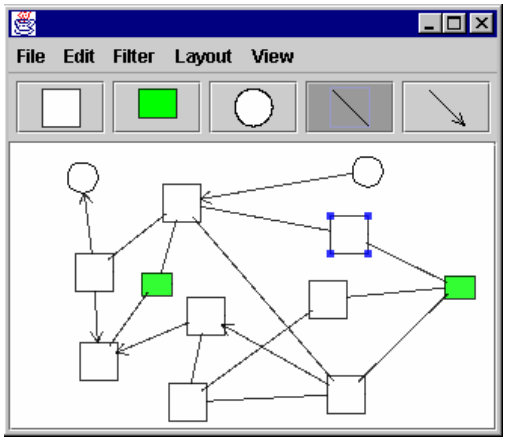


Figure 6. A prototype system for graph layout

We can see different layout results with different seed nodes for the graph in Figure 7: the graph in Figure 7(b) is generated by using the leftmost node as the seed node in FTA, and the graph in Figure 7(c) using the centre node as the seed node. If we want to find which node in the graph is the best candidate for the seed node, we should try every node to calculate the cost function and find the minimum one. Our prototype system provides a facility for the user to use the mouse to select the seed node for a graph; this would be faster than using the cost function. For example, for the graph in Figure 8 (a), when the user selects the central position as the dummy seed node, the layout result will be the graph in Figure 8 (b), which is better that that in Figure 8(c) with the leftmost node as the seed node, because only two nodes in Figure 8(b) are moved in the left or right direction, while three nodes in Figure 8(c) are moved in the right direction.

Figure 9 (b) and Figure 9 (c) show that the forces are not transferred from one cluster node to another by using FTA, because the formula (2.5) is not satisfied in this graph. In the bottom level of the graph in Figure 9(a), there are two overlapping nodes. When the force is used to remove the overlap, the force is not transferred to other nodes on the right of the first overlapping node in Figure 9(a).

Figure 10, 11 and12 show the initial and resulting layouts for graph 4, 5 and 6, respectively. The graphs adjusted by FTA are more compact than those by FSA. Furthermore, the areas of the resulting layouts by FTA are smaller than those by FSA.

The biggest node Graph 7 in Figure 13(a) overlaps with other two nodes, so FTA moves only the biggest node in Figure13 (b). However FSA moves all the nodes located on the right of the first node that is overlapping with the biggest node.

Graph 8 in Figure 14 is used to show that layout adjustment may introduce edge-node intersections by using FSA, although FTA does not produce edge-node intersections for this graph. However, sometimes FTA may also generate edge-node intersections. Fortunately, this problem can be easily solved by the approach in [1].

Graph 9 in Figure15 is an X-shaped graph with symmetry about both the x- and y-axis. Both layouts by FTA and FSA are reasonable but the gaps between nodes in Figure 15(c) are a little bigger.

Graph	3	4	5	6	7	8	9
Nodes	10	13	8	12	10	5	18
FTA	7	127	86	76	4	18	258
FSA	74	1318	457	399	89	85	1335

Table 1 Value of Cost Function $f(v_1)$

In summary, we conclude that graphs adjusted by FTA are more compact than those by FSA from above examples. Furthermore, we can also use a numerical measure of the quality of the adjusted layout by the cost function previously defined. The results are reported in Table 1, which shows the values of cost function $f(v_1)$ of using FTA and FSA for those graphs in Figure 9 – Figure 15 (i.e. Graph 3 – Graph 9). They can also be illustrated in Figure 16. Note that the leftmost nodes (v_1) of Graph 3 – Graph 9 are chosen as the seed nodes for the cost function.

In [5], the authors point out that the ranking of methods in terms of CPU time taken to find the adjusted layout, from fastest to slowest is Uniform Scaling, Force-Scan, and

Constrained Optimization. Uniform scaling and Force-Scan are considerably faster than Constrained Optimization. Our new approach, the Force-Transfer, has the same time complexity as the Force-Scan. It is not only a fast algorithm for removing overlapping nodes but also make the layout more compact.

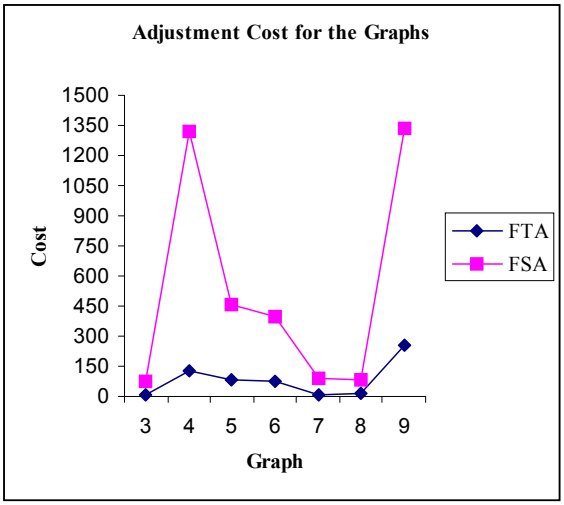


Figure 16. Comparison of values of Cost function for Graph 3 - Graph 9

6. Conclusion

In this paper, we propose a new approach to removing overlapping nodes in graph layout. Compared with FSA and other algorithms, we conclude that FTA is a good choice of balancing between a nice layout and less calculation time. Our approach is not only used to adjust graphs, but it can also be applicable to lay out non-overlapping objects, such as windows and labels, by specifying a minimum gap between these objects.

Acknowledge We would like to thank Dr. Gitesh K. Raikundalia for his comments on this paper.

References

[1] WEI LAI and PETER EADES(2002): Removing edge-node intersections in drawings of graphs, *Information Processing Letters* , **81**:105-110.

[2] WEIQING He and KIM MARRIOTT(1998): Constrained Graph Layout, *Constraints*, **3** (4): 289-314.

[3] G.DI BATTISTA,P.EADES,R.AMASSIA and I.TOLLIS(1998):*Graph drawing: algorithms for the visualization of graphs*, Prentice Hall.

[4] P.EADES and WEI LAI(1992): Algorithms for disjoint node images. *In Processings of the 15th Australian Computer Science Conference*, 253-265, Hobart.

[5] KIM MARRIOTT, P.STUCKEY,V.TAM and W.HE: Removing Node Overlapping in Graph Layout Using Constrained Optimization, *Constraints*, 1-31.

[6] W.HE and K.MARRIOTT(1998): Removing node overlapping using constrained optimization, *In Processing of the 21st Australian Computer Science Conference*, Perth, 169-180.

[7] WEI LAI(1993): *Building Interactive Diagram Applications*. PhD thesis, Department of Computer Science, University of Newcastle.

[8] R. DAVIDSON and D. HAREL(1996): Drawing Graphs Nicely Using Simulated Annealing. *ACM Transactions on Graphics*, **15**(4):301-331.

[9] B.P. BUCKLES and F.E. PETRY (1992): *Genetic Algorithms*, Computer Society Press, Los Alamitos, CA.

[10] GLOVER, F.(1989): Tabu Search, Part I , *ORSA Journal on Computing*, **1**(3):190-206.

[11] GLOVER, F.(1989): Tabu Search, Part IT, *ORSA Journal on Computing*, **2**(1):4-32.

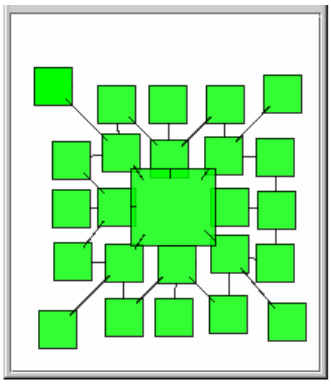
[12] P.EADES,W.LAI, K.MISUE and K, SUGIYAMA (1995): Layout adjustment and the mental map, *J. Visual Languages Comput*,**6**:183-210.

[13] KUNIHIKO H., MICHILO I.,TOSHIMITSU M., and HIDEO F. (2002):A layout adjustment problem for disjoint rectangles preserving orthogonal order, *Systems and Computers in Japan*, **33**(2):31-42.

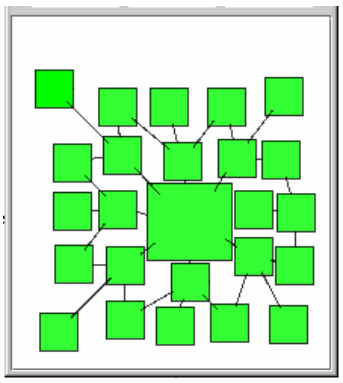
[14] K.A.LYONS, H.MEIJER, and D.RAPPAPORT (1998): Algorithms for cluster busting in anchored graph drawing. *Journal of Graph Algorithms and Applications*, **2**(1):1–24.

[15] M.D.STORY and HAUSI A. MüLLER(1995): Graph layout adjustment strategies. *In Symposium on Graph Drawing, GD'95*, LNCS 1027, 487-499, Passau Germany, September, Springer-Verlag.

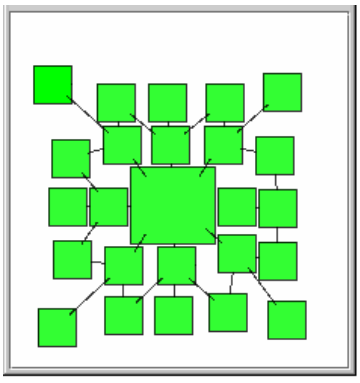
[16] NATASA MILIC-FRAYLING and RALPH SOMMERER(2002): SmartView: Flexible Viewing of Web Page Contents, *the Eleventh International World Wide Web Conference*, Hawaii, USA.



(a) Graph 1

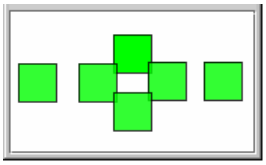


(b) Layout by FTA with Vs=V1

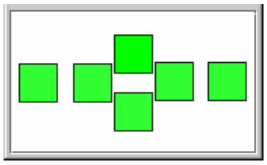


(c) Layout by FTA with Vs=centre node

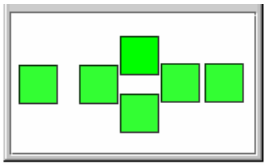
Figure 7. Initial and adjustment layouts for graph 1



(a) Graph 2

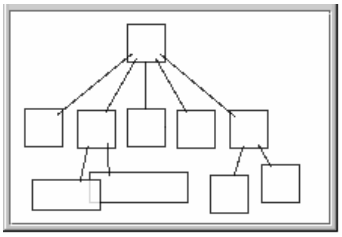


(b) Layout by FTA with a virtual node in the centre

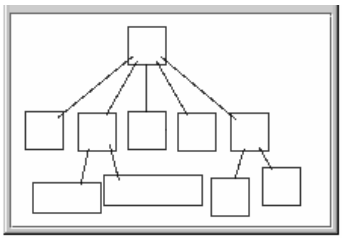


(c) Layout by FTA with v_s=v_1

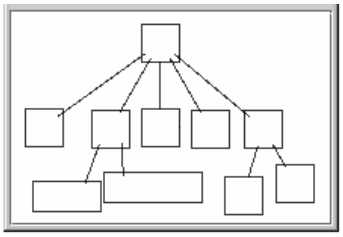
Figure 8. Initial and adjustment layouts for graph 2



(a) Graph 3

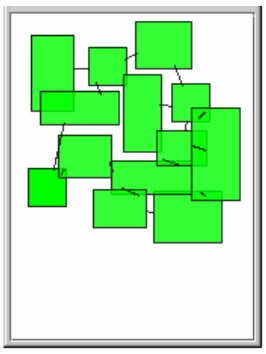


(b) Layout with FTA

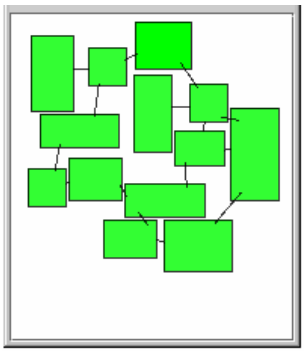


(c) Layout with FSA

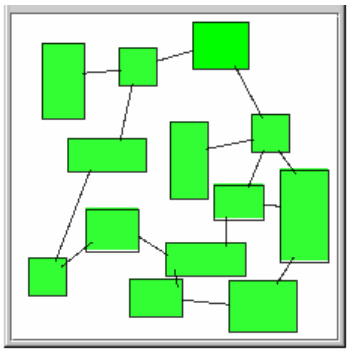
Figure 9. Initial and adjustment layouts for graph 3



(a) Graph 4

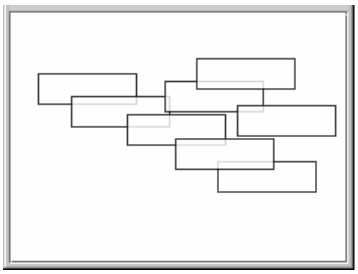


(b) Layout with FTA

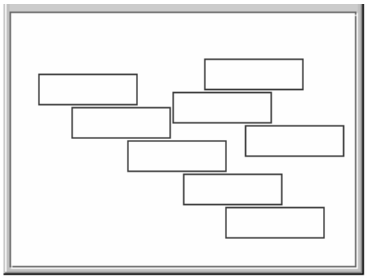


(c) Layout with FSA

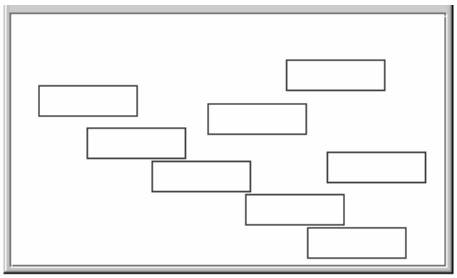
Figure 10. Initial and adjustment layouts for graph 4



(a) Graph 5

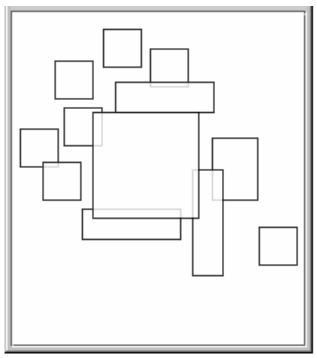


(b) Layout with FTA

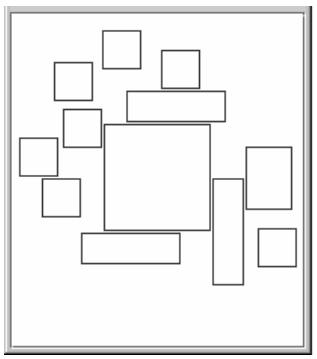


(c) Layout with FSA

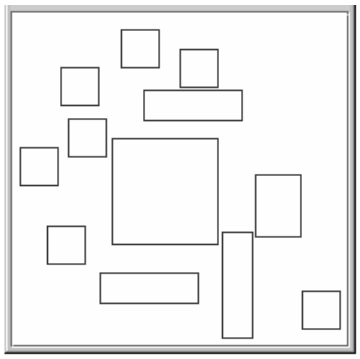
Figure 11. Initial and adjustment layouts for graph 5



(a) Graph 6

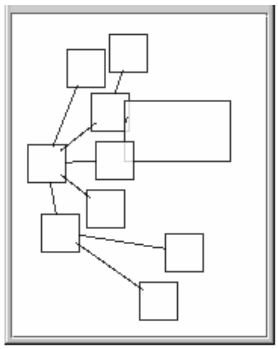


(b) Layout with FTA

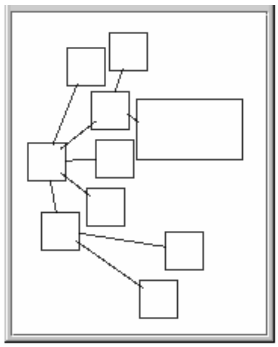


(c) Layout with FSA

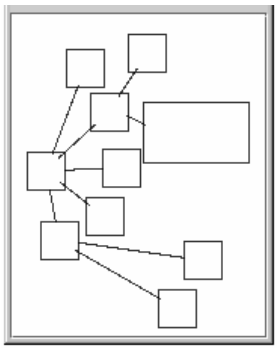
Figure 12. Initial and adjustment layouts for graph 6



(a) Graph 7

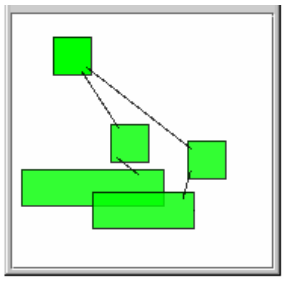


(b) Layout with FTA

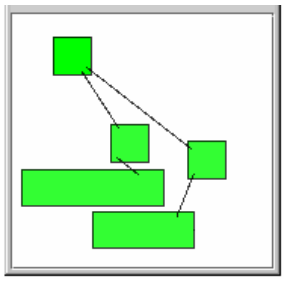


(c) Layout with FSA

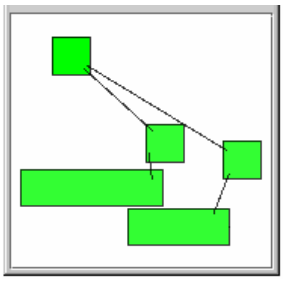
Figure 13. Initial and adjustment layouts for graph 7



(a) Graph 8

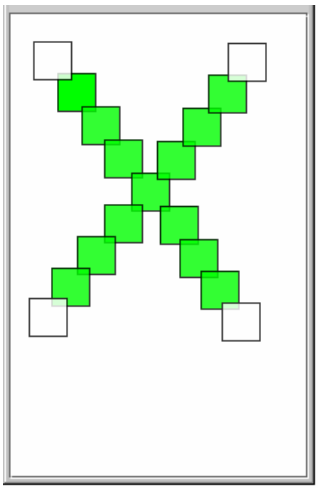


(b) Layout with FTA

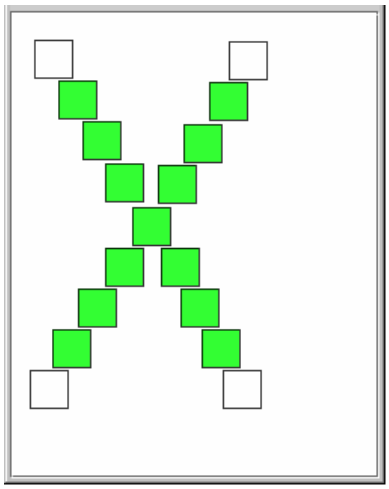


(c) Layout with FSA

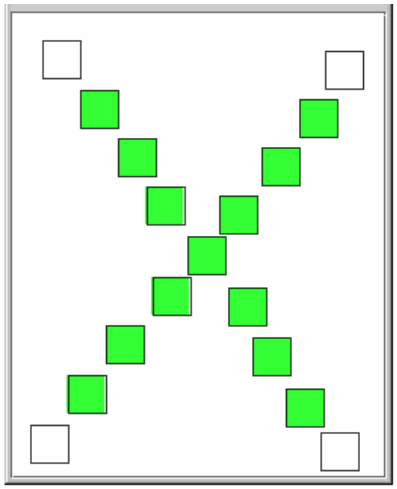
Figure 14. Initial and adjustment layouts for graph 8



(a) Graph 9



(b) Layout with FTA



(c) Layout with FSA

Figure 15. Initial and adjustment layouts for graph 9