Mental Map Preserving Graph Drawing Using Simulated Annealing

Yi-Yi Lee, Chun-Cheng Lin, Hsu-Chun Yen*

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan 106, R.O.C.
E-mail: {sthugo,sanlin}@cobra.ee.ntu.edu.tw, yen@cc.ee.ntu.edu.tw

Abstract

Information visualization has attracted much attention in recent years in many fields of science and engineering. In many applications, graphs are 'dynamic' in the sense that changes are constantly applied to a graph to reflect the evolution of the system behaviour represented by the graph. In the past, the concept of the so-called "mental map" has largely been ignored. Users often have to spend a lot of time relearning the redrawn graphs. This paper proposes an effective way to release the user from such kind of a distasteful job by maintaining a high degree of the "mental map" for general graphs when graphs are redrawn. Our experimental results suggest this new approach to be promising.

Keywords: mental map, simulated annealing, graph drawing.

1 Introduction

Using graphs to represent real-world concepts has been widely used in many areas such as social relationship, interactive systems, database, automata theory, and more. In many cases, users may interact with the graphs, such as adding/deleting one or more nodes/edges, on a regular basis. As drawing graphs manually becomes infeasible for graphs of high complexity in structure and size, many automatic graph drawing methods have been developed. Most of the existing automatic drawing methods draw graphs according to a pre-built model. Such drawing methods are convenient and simple but have some limitations. Since only a few methods maintain the previous information which was formed in the users' mind after they read the graphs, these techniques cannot preserve the users' "mental maps". The redrawn graph may differ a lot in structure in comparison with the original drawing, even in the case when only a slight modification was made to the original graph. When this happens, the user often has to spend a considerable amount of extra time to recognize the new graph. When the frequency of modification increases, this problem becomes more serious.

* Research supported in part by NSC 94-2213-E-002-086, Taiwan.

Suppose Figure 1(a) is the drawing of a graph using the well-known spring algorithm. After adding node 4 and connecting it to the rest of the graph, running the spring algorithm produces the layout displayed in Figure 1(b). Clearly Figure 1(c) is a better one for preserving the mental map since it inherits more information from the previous graph (i.e., Figure 1(a)) in comparison with Figure 1(b). Figure 2, from (Kaufmann and Wiese, 2002), is another example illustrating the importance of the preservation of the mental map. In Figure 2, adding a single edge is going to harm the mental map considerably. The initial graph consists of a chain of quadrangles, which are displayed nicely by the layout algorithm "Circular with Radial" (see (Kaufmann and Wiese, 2002)). When we connect the two outermost quadrangles by a single edge which is marked by the thick curve connecting from the highest node to the lowest node as shown in Figure 2(a), all of the components collapse into one, as shown in Figure 2(b). The Circular with Radial algorithm places the vertices on a circle and then performs a crossing minimization heuristics, which still results in at least $n/4$ crossings, one from each of the quadrangles. Unfortunately, the new drawing suffers from a rather small angular resolution such that the previous structure can hardly be recognized. An alternative layout (Figure 2(c)) for this graph shows the complete ring of components as it arose from the construction. When the graph size grows, the problem of then mental map will be more critical.

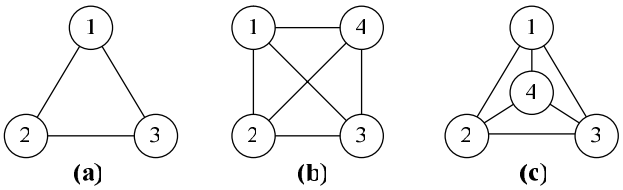


Figure 1. An example of the mental map problem.

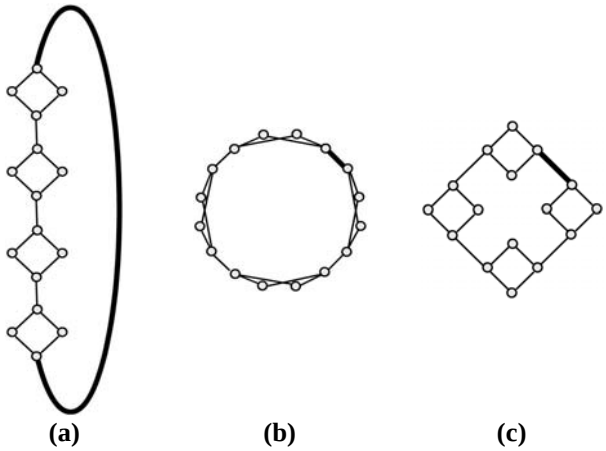


Figure 2. Example of the mental map problem in circular drawing (Kaufmann and Wiese, 2002).

The *mental map* concept was first proposed in (Eades et al., 1991). *Mental map* can be thought of as the presentation of a person's mind. When one sees a graph, there is a mental map constructed in his mind. Some people refer to the problem of preserving the user's mental map as the stability problem, see, e.g., (Paulisch and Tichy, 1990).

Preserving the mental map allows the user to quickly recognize the new graph which is redrawn after a modification is made to the original graph. The existing algorithms solving the mental map problem mostly focus on maintaining the orthogonal ordering, such as (Misue et al., 1995), or adding constraints to the graph, such as (K. Böhringer et al., 1990), and (He and Marriott, 1998). Others approaches preserve the mental map on certain kinds of graph layouts, see, e.g., (Kaufmann and Wiese, 2002). To our understanding, solving the mental map problem for general graphs under more sophisticated criteria remains a challenge.

Drawing graph nicely means placing nodes and edges in a pleasing way. There are different aesthetic criteria in different areas. In (Davidson and Harel, 1996), a nice-looking graph drawing refers to a drawing which: (1) distributes nodes evenly, (2) makes edge-lengths uniform, (3) minimizes edge-crossings, and (4) keeps nodes from coming too close to edges. To draw graphs nicely, a simulated annealing method was used in (Davidson and Harel, 1996) with the cost function designed to respect the above requirements. The implementation of their work allows the user to adjust the relative weights among various parameters, although some of these criteria may be in conflict with each other. Simulated annealing offers a more flexible way of drawing graphs than many other algorithms. The criteria used in (Davidson and Harel, 1996) will be referred to as '*nice criteria*' throughout the rest of this paper.

There are many intuitive factors that affect the mental map. For example, (Bridgeman and Tamassia, 2002) formally defined those intuitive factors and presented some statistical results to evaluate their effects on human perception of similarity. They used many measurements to measure the similarity between different pictures, and then asked users to measure the similarity between the same set of graphs. Finally, they used a statistical method to find the relationship between these measurements and users feelings, and concluded the best of the measures evaluated as follows: *ranking, orthogonal ordering, λ-matrix, nearest neighbour within, average distance, weighted nearest neighbour between, shape*, and *relative distance*.

In this paper, we propose a mental map preserving graph drawing algorithm for general graphs, based upon the simulated annealing graph drawing algorithm of (Davidson and Harel, 1996) with the cost function incorporating six criteria of (Bridgeman and Tamassia, 2002) to reflect people's mental maps. Our evaluation is divided into three stages. The first stage uses the original simulated annealing approach to draw graphs. The second stage is to modify the graph slightly. In most cases, we add two nodes, one with two edges and the other with four edges. In the third stage, our new approach is applied to redraw the modified graph nicely while preserving the mental map. The experimental results suggest this new approach to be promising. To a certain extent, our approach can help us preserve the contour and the relative positions of the drawing of a graph when a sequence of operations are applied. Similar to (Davidson and Harel, 1996), our approach allows the user to adjust the relative weights among various parameters, and hence is very flexible. In addition, we can see from the experimental results that there exists a trade-off between drawing graphs nicely and preserving mental map.

The contributions of this paper are summarized as follows:

1. The previous work of preserving the mental map focused on how to preserve the orthogonal ordering and adding additional constraints to the graph. To our understanding, no graph drawing algorithms aiming at preserving the mental map on general graphs while using more sophisticated criteria simultaneously have been reported in the literature.

2. Our approach allows the user to adjust the relative weights among various parameters, and hence is very flexible.

3. By taking samples among a group of students with basic knowledge of graphs and graph drawing, (Bridgeman and Tamassia, 2002) suggested a number of good criteria suitable for capturing the notion of the mental map. Our graph drawing approach incorporates the criteria of (Bridgeman and Tamassia, 2002) into the cost function of a simulated annealing algorithm to preserve the contour and relative positions of the drawing of a graph when a sequence of operations are applied.

1. Because our approach obtains the location of each node in each iteration, rendering the graph in each iteration results in a smooth process from the initial drawing to the final drawing – a desirable property in information visualization.[1]

The rest of this paper is organized as follows. In Section 2, we introduce the simulated annealing method on which our graph drawing algorithm is based. Section 3 involves the design of the cost function to maintain the mental map. Section 4 shows the experimental results of our algorithm. Concluding remarks and future work are given in Section 5.

2 Drawing Graphs Nicely

As mentioned in the introduction, the four requirements of a "*nice drawing*" include distributing nodes evenly, making edge-lengths uniform, minimizing edge-crossings, and keeping nodes from coming too close to edges.

2.1 Introduction to Simulated Annealing

The algorithm presented in (Davidson and Harel, 1996) was based on the technique of *simulated annealing* (SA), a generalization of the Monte Carlo method for examining the equations of state and frozen states of *n*-body systems.

[1] The experimental results are not given in this paper because the application to a smooth process is intuitive.

This method simulates the freezing of liquids or the recrystallization of metals in the process of annealing. In the beginning the system is at high temperature and disordered. As the process proceeds, the temperature cools down and the system becomes ordered.

SA is flexible and has successfully been applied to solving various combinational optimization problems. It allows "uphill" moves — moves that spoil, rather than improve, the temporary solution. This unique feature allows SA to escape from a local minimal solution, although there is no guarantee that a global minimum can be reached eventually.

The first step of SA is to generate an initial configuration (either randomly or heuristically constructed) and initialize the so-called temperature parameter *T*. Then the following is repeated until the termination condition is satisfied. A new solution from the neighbourhood of the current solution is randomly selected. The energy cost of the new solution is calculated over the objective function and *T* is decreased. Let E and E' denote the current and new energy costs, respectively. If $E' < E$, the new solution will be accepted. If $E' > E$, this solution will be accepted with a probability which is a function of *T* and $(E' - E)$. This probability is from the Boltzmann distribution: $e^{-(E'-E)/kT}$ where *k* is the Boltzmann constant. (We assume *k* = 1 in the subsequent discussion.) This kind of "uphill" moves will be fewer as SA proceeds because *T* is decreased after each iteration.

The following are the basic steps of SA:

1. generate an initial configuration δ and an initial temperature *T*;
2. repeat the following until the termination condition is satisfied:
   (1) choose a new configuration δ' from the neighbourhood of δ;
   (2) let E and E' be the value of the cost function at δ and δ' respectively; if $E' < E$ or random $< e^{-(E'-E)/kT}$ then set $\delta \rightarrow \delta'$;
3. decrease the temperature *T*;
4. if the termination condition is satisfied, stop; otherwise go to step 2.

2.2 The Graph Drawing Algorithm

The input to our algorithm is a pair (*V*, *E*) stored in an adjacency list, where *V* is the set of nodes and *E* is the set of edges. Each node in *V* has a unique index. The output of our algorithm is a graph drawn on an $n \times n$ rectangular grid, where *n* depends on the size of the graph. The following is the setting of SA needed to carry out the simulation of the physical annealing process:

A *configuration* is the placement of nodes and edges of the graph. If there is no prior information about the graph, we place the graph randomly. The *neighbourhood* of a configuration *C* is a configuration that is obtained from *C* by changing the location of a single node. In each iteration, we choose a node and then randomly assign a new location to it. In the beginning, the distance between a configuration and its neighbourhood is very large. As the algorithm continues, the distance between two consecutive locations of a node decreases. This setting can accelerate the convergence of the algorithm.

2.2.1 The Cost Function

The *cost function* constitutes the most important part of SA, on which the efficiency of the algorithm highly depends. A good cost function should reflect the properties of the nice-looking pictures. Intuitively, all desired features of the final picture must contribute to this function, in accordance with their relative importance. The following are the criteria that we use in the design of the cost function. These criteria are intuitively clear, using which nice drawings of graphs are obtained.

(1) *Node distribution*: This criterion is concerned with how to spread the nodes of a graph evenly. In a nice drawing, any two nodes should not be too close to each other. Nodes should not be overcrowded and should be evenly distributed in the drawing area. These intentions can be accomplished by the "*Node Distribution*" and "*Borderlines*" components of the cost function. The former deals with the distances between any two nodes, while the latter deals with the distances between a node and four borderlines of the space. For each pair of nodes *i* and *j*, the term:

$$\alpha_{ij} = \frac{\lambda_1}{d_{ij}^2}$$

is added to the cost function, where d_{ij} is the Euclidean distance between nodes *i* and *j* and λ_1 is a normalizing factor that defines the relative importance of this criterion compared to others. Increasing λ_1 relative to other normalizing factors will produce a smaller graph.

(2) *Borderlines*: This component prevents nodes from coming too close to the borderlines. This element and *Node Distribution* can disperse the nodes evenly. For each node *i*, the term is added to the cost function:

$$m_i = \lambda_2 \left(\frac{1}{r_i^2} + \frac{1}{l_i^2} + \frac{1}{t_i^2} + \frac{1}{b_i^2}\right)$$

where r_i, l_i, t_i and b_i stand for the distances between node *i* and the right, left, top and bottom sides, respectively; λ_2 is the normalizing factor, and increasing λ_2 relative to other normalizing factors will make the graph towards the center of the drawing space.

(3) *Edge lengths*: The previous two ingredients are related to nodes. Now we discuss the criteria about edges. This component is slightly different from that used in (Davison and Harel, 1996). They used the term, $c_k = \lambda_3 d_k^2$ where d_k is the length of edge k. In our setting we use the term

$$c_k = \lambda_3 s^2$$

where *s* is the standard deviation of edge lengths and λ_3 is the normalizing factor. The reason why we use standard deviation instead is that using only edge length in designing the cost function tends to produce smaller graph layouts that are undesirable in some cases. Our new criterion ensures that edge lengths do not vary too much, though this alternative criterion costs more time than using that proposed in (Davison and Harel, 1996).

(4) *Edge crossings*: Minimizing the number of crossing is an important goal, but it is in general difficult to achieve. Too many crossings degrade the quality of a drawing, which often cause unnecessary confusions when people reading the graph. Although the output of our SA approach may not reach the minimal number of crossings theoretically, it produces a quite pleasing picture in most cases as our experimental results indicate. To incorporate this component we simply add to the cost function the term:

$$e = \lambda_4 f_k f_l$$

where f_k is the weight attributed to edge *k*, again λ_4 is a normalizing factor that defines the relative importance of this criterion compared to others. In this case, the normalizing factor, λ_4, is defined as

$$\lambda_4 = \frac{\lambda_5}{g_{\min}^2}$$

where λ_5 is the normalizing factor of the next component, *Node-edge Distance*, and $g_{\min}$ is the minimal distance between all nodes and edges in the current configuration. This component constitutes the most time consuming one among all terms in the cost function.

(5) *Node-edge distances*: The *Edge Crossing* component penalizes edges that cross with other edges, but does not penalize the condition in which nodes become too close to neighbouring edges. Nodes that are too close to their neighbouring edges are likely to confuse the user. So for node *k* and edge *l* with distance g_{kl}, the value of

$$h_{kl} = \frac{\lambda_5}{g_{kl}^2}$$

is added to the cost function. λ_5 is the normalizing factor, and lager λ_5 prefers longer distances between nodes and edges.

2.2.2 The Cooling Schedule

The cooling schedule is one of the key parts of the annealing algorithm. Most applications follow the basic geometric schedule described in (van Laarhoven and Aarts, 1987).

1. *Initial temperature:* Generally, we set the initial temperature high enough to accept almost every move in the beginning, in order for any configuration to work as the starting point of the algorithm. In some situations, however, when the initial configuration is known to have some resemblance to the desired final result, a lower initial temperature can be chosen at the beginning..
2. *Temperature reduction:* The *temperature reduction* stage determines when to change the temperature and how to change it. This stage affects the number of moves at each temperature and the range of each move. Typical applications set the number of trials at each temperature to be polynomial in the size of the input. We set the moves at each temperature to be 30 times the number of nodes. The cooling schedule rule is geometric. If T_p is the temperature at the *p*-th stage, then the temperature at the next stage is

   $$T_p = \gamma T_p,$$

   with γ is a real number between 0.6 and 0.95. The range of each move decreases when the temperature reduces. Given the range of movement at the *p*-th stage is *R*; the range of movement serving for the next temperature will be γR, which accelerates the convergence of the algorithm.
3. *Termination condition:* This part is to decide when to stop the algorithm. Commonly used termination conditions include: stopping the algorithm when the solution does not change after several consecutive stages, or fixing the number of stages to be a constant. Evidence has shown that, in many cases, substantially changes happen in the first few stages and the following stages yield marginal improvement.

3 Maintaining the Mental Map

When a user looks at a graph, he or she learns about the graph's structure, and subsequently navigates through the drawing to understand its meaning. This is what we called the "*mental map*." This idea can be used in many areas. Since there are many real-world concepts captured by means of graph representations (such as flow-chart, social relationship, automata, …, etc), preserving the user's mental map from the existing information is getting more and more important as the size of the graph increases.

In an interactive system, users or applications may modify the graph very often. As we explained earlier, adding one node and three edges to Figure 1 may cause some confusion. If we make more changes, the situation is likely to get worse. Although animation can be used to provide a smooth transition between consecutive drawings, it is still important to preserve some degree of similarity between drawings. If we cannot maintain one's mental map, he or she may have to spend a lot of time relearning the new graph.

In the past, maintaining the mental map in graph drawing has only been scarcely studied. Most of these works focused on how to preserve the orthogonal ordering and adding additional constraints to the graph. To our understanding, no algorithm aiming at preserving the mental map on general graphs while using more sophisticate criteria simultaneously has been reported in the literature. We feel that orthogonal ordering is not the only factor affecting one's mental map. Seeking additional

factors regarding the preservation of the mental map plays an important role in drawing, for example, dynamic graphs. Furthermore, adding additional constraints by the user becomes infeasible as the size of the graph increases. Regarded by many as the most complete work about the measurements of the mental map, Bridgeman and Tamassia (2002) delivered a user study based on several similarity measures, which reflect one's mental map according to the direct responses of users.

3.1 The Framework of Bridgeman and Tamassia

Bridgeman and Tamassia (2002) used a statistical method to test the mental map. Their samples include 103 students in a CS course at Brown University. Before taking the test the students had eight lectures on graphs and graph algorithms, including one on graph drawing. They had also been assigned a programming project involving graphs, so they had some familiarity with the subject. Though the study was done on orthogonal graphs, these measurements perform equally well on general graphs.

Each graph in the experiment was produced by Giotto and InteractiveGiotto. These graphs can be divided into two categories: base drawings and modified drawings. The base drawings were drawn by Giotto, while the modified drawings were produced by InteractiveGiotto, which simulated a user's action by adding two nodes, one with two edges and the other with four edges. InteractiveGiotto preserves edge crossings and bends.

In order not to affect the correction of the experiment, students were allowed to take a break after each task; so they did not have to stay focused for too long without a break. At each part students were asked to decide which graph resembles the base drawing or to find out which vertex in the modified graph is not in the base drawing. The choices as well as the response time the students made were recorded. It was assumed that the more time students spent, the less similar these graphs were. If the graphs were too different for the users to make a choice, they could choose the "cannot decide" answer. On the other hand, if the users spent less time answering a task, then the modified graph in the task tends to be more similar to the base graph.

Based on the above experiment, Bridgeman and Tamassia (2002) suggested a number of measurements that are best suited for capturing a user's mental map. We choose these criteria to enrich our cost function.

3.2 Redrawing Graphs while Preserving the Mental Map

In this stage, we use the simulated annealing method to draw graphs. Most of the settings in this stage are the same as those discussed in the previous section. We only explain the differences. The value of each component of the cost function is nonnegative. Given a value of a component of the cost function *M*, *M* = 0 implies that these two graphs are thought to be the same according to this component. The larger magnitude of *M* means a higher degree of difference between the graphs.

3.2.1 The Cost Function

To maintain the mental map, we add several criteria to our cost function. In addition to the components mentioned previously, we enrich the cost function with the following components. These components are designed to reflect the critical factors regarding the preservation of the mental map based on the experiments reported in (Bridgeman and Tamassia, 2002). In the following, P and P' refer to the point sets for graph layouts D and D', respectively, and $p' \in P'$ is the corresponding point for $p \in P$ (and vice versa). Let $d(p,q)$ be the Euclidean distance between points *p* and *q*.

1. *Rankings*: Considering the fact that the relative horizontal and vertical positions of a node should not vary too much, we add the component, *ranking*, to the cost function. Let *right(p)* and *above(p)* be the numbers of points to the right and above of *p*, respectively. λ_6 is the normalizing factor.

$$rank(P,P') = \frac{1}{\lambda_6}\sum_{p\in P}\min\{|right(p)-right(p')| + |above(p)-above(p')|\}$$

where $\lambda_6 = 1.5(|P|-1)$.

2. *Average relative distance*: The *average relative distance* is the average of the distances between pairs of nodes. Users normally do not want to see the distances between nodes in the redrawn picture to vary too much because too many changes on the distances would destroy the mental map.

$$rdist(P,P') = \frac{1}{|P|(|P|-1)}\sum_{p,q\in P}|d(p,q)-d(p',q')|$$

3. *Shape*: The *Shape* factor records the orientation of each edge of the graph as one of the following directions: north, south, east, west, north-east, south-east, north-west and south-west. By keeping the relative positions of any two nodes, we can maintain a better mental map. λ_7 is the normalizing factor.

$$shape = \frac{1}{\lambda_7}\sum_{e\in E} edits(e,e')$$

where $\lambda_7 = |E|$ and $edits(e,e') = \begin{cases}1, & \text{if } e \neq e' \\ 0, & \text{otherwise}\end{cases}$

in which e' is the corresponding edge of e in D'.

4. *λ-matrix*: Two sets of points *P* and *P'* have the same order type if, for every triple of points (*p*, *q*, *r*), they are oriented counter-clockwise if and only if (p',q',r') are also oriented counter-clockwise. The λ-matrix is designed to maintain the kind of an order type. By using a simplified version, we calculate the number of nodes left to two designated nodes. Let $\lambda(p,q)$ be the number of points in *P* to the left of the directed line from *p* to *q* and *n* be the size of *P*. λ_8 is the normalizing factor.

$$lamda(P,P') = \frac{1}{\lambda_8} \sum_{p,q \in P} |\lambda(p,q) - \lambda(p',q')|$$

where $\lambda_8 = n\left\lfloor \frac{(n-1)^2}{2} \right\rfloor$.

5. *Nearest neighbour within*: This component captures the intuitive idea that a node's nearest neighbour should remain the closest one to it after the graph is redrawn. Let *nn*(*p*) be the nearest neighbour of *p* in *P* and *nn*(*p*)' be the corresponding node in P' to *nn*(*p*). λ_9 is the normalizing factor.

$$nnw(P,P') = \frac{1}{\lambda_9} \sum_{p \in P} nearer(p)$$

where $\lambda_9 = |P|$ and $nearer(p) = \{q \mid d(p',q') < d(p',nn(p)'), q \in P, q \neq p, q \neq nn(p)\}$

6. *Nearest neighbour between*: This concept is based on the assumption that a node should not move too far from the original position. And its new location should be the closest one to its previous location. When the new drawing is presented, the user ought to be able to find a certain node according to the mental map. λ_{10} is the normalizing factor.

$$nnb(P,P') = \frac{1}{\lambda_{10}} \sum_{p \in P} nearer(p)$$

where $\lambda_{10} = |P|(|P|-1)$ and $nearer(p) = \{q \mid d(p,q') < d(p,p'), q \in P, q \neq p\}$

Figure 3 is the overview of our algorithm.

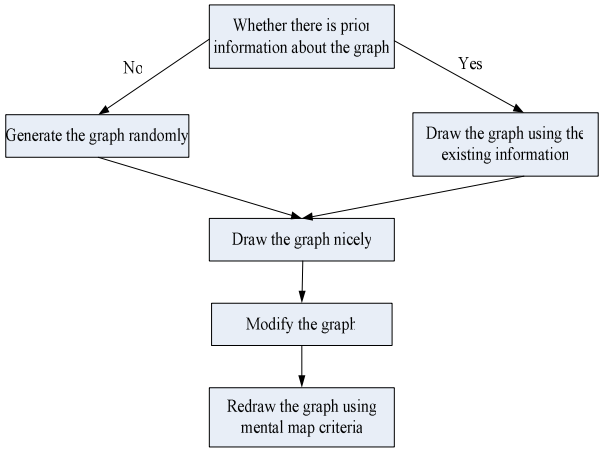


Figure 3. The overview of our algorithm.

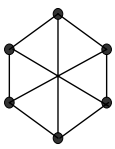
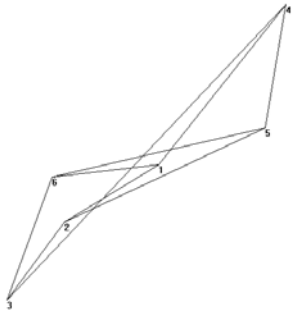
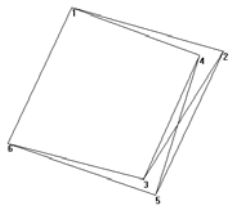


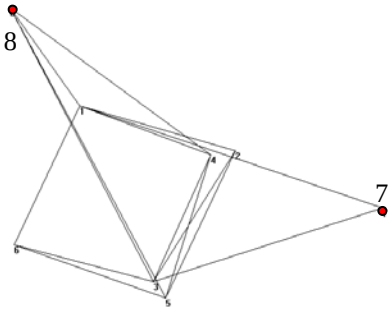
Figure 4. An example.



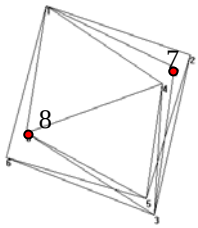
(a) the initial drawing.



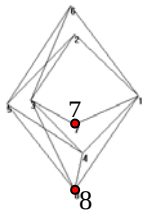
(b) applying "nice criteria" to (a).



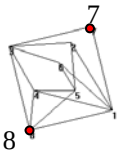
(c) adding nodes 7 and 8.



(d) applying "mental map criteria" to (c).

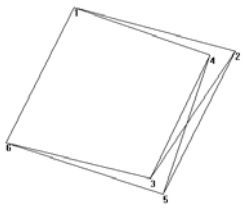


(e) applying "nice criteria" to (c).



(f) applying "nice criteria" to a random drawing of the graph of (c).

Figure 5. Various drawings illustrating the preservation of the mental map.



(a) Figure 5(c) without nodes 7, 8.

(b) Figure 5(d) without nodes 7, 8.

(c) Figure 5(e) without nodes 7, 8.

(d) Figure 5(f) without nodes 7, 8.

Figure 6. Removing nodes 7 and 8 from Figures 5(c)–(f).

4 Experimental Results

This section shows a number of examples to which our mental map preserving drawing algorithm is applied. We first draw a graph using the "*nice criteria*" explained in Section 2. Like the experiments carried out in (Bridgeman and Tamassia, 2002), we add two nodes, one with two edges and the other with four edges. The locations of these nodes are chosen randomly. The connections between the added nodes and the original graph are decided randomly. Finally, we redraw the modified graph using our algorithm which incorporates the "*mental map criteria*" discussed in Section 3. For a comparison purpose, we also present the modified graphs which are drawn using the "*nice criteria*" only.

Our prototype system runs on a Pentium IV 2.4 GHz PC with 512 MB memory. Consider a very simple example shown in Figure 4. In our implementation, a random drawing was generated as shown in Figure 5(a). After applying the "*nice criteria*", the SA-based drawing algorithm produced the graph displayed in Figure 5(b). Then two nodes, 7 and 8, were added whose locations as well as the connections to the remaining graph were decided randomly (see Figure 5(c)). The modified graph was redrawn by using our modified SA approach based on the "*mental map criteria*." The result is shown in Figure 5(d). The running time required to generate Figure 5(d) is 0.172 second. To compare with drawings without taking into account the concept of mental map, we show two additional drawings. Figure 5(e) shows the drawing after applying the "*nice criteria*" to Figure 5(c). Figure 5(f) shows the drawing based on the "*nice criteria*" with respect to a randomly generated initial drawing of the modified graph.

In view of Figures 5(d) and 5(b), it is easy to see that their contours and relative positions are very similar. On the other hand, Figures 5(e) and 5(f) share little similarity with Figure 5(c). Further, if we delete the additional nodes 7 and 8 from the drawings displayed in Figures 5(c)-(f), it becomes clearer why Figure 5(d) preserves a better mental map (see Figure 6). Like many aesthetic criteria which are often in conflict with one another, there is a trade-off between maintaining a high degree of mental map and other criteria such as minimizing the number of crossings. Our algorithm allows users to adjust the weights between different criteria, in order to fit their drawing requirements.

This next example is a tree-like graph with 21 nodes and 20 edges shown in Figure 7. With respect to the initial drawing shown in Figure 8(a), Figure 8(b) displays the output drawing using the 'nice criteria' algorithm. Figure 8(c) shows the graph after nodes 22 and 23 (together with some edges) are inserted into the graph. Note that in Figure 8 (c), only node 19 was pulled upper, node 21 was pulled lower and some edges with the same parent changed their relative positions. Figure 8(d) is the drawing produced by applying our mental map preserving algorithm to Figure 8(c). There are no substantial differences between Figures 8(b) and 8(d). Figures 8(e) and 8(f) are drawings obtained from Figure 8(c) without taking the mental map property into account. Although Figure 8(c) is a planar graph, Figure 8(d) displays two crossings; however, there are no crossings in Figure 8(e) and 8(f). Like we mentioned earlier, preserving the mental map and achieving optimality with respect to some other aesthetic criteria may not be met simultaneously. Figure 9 simply displays the drawings after the additional nodes 22 and 23 are removed from Figures 8(c)-(f). Figure 9(b) still preserves the shape and structure of Figure 9(a). In contrast, Figures 9(c) and 9(d) look somewhat different from Figure 9(a).

Finally, we present a tree example which has 63 nodes and 62 edges. It takes 39.313 seconds to finish the drawing which is based on the 'nice criteria'. Using the mental map criteria, the drawing takes 45.266 seconds. See Figure 10. See also Figure 11 which shows the drawings when the added nodes are removed from Figure 10. It is fairly easy to see the advantage of our drawing algorithm when preserving mental map is the primary concern in drawing graphs.

5 Conclusion and Future Work

In this paper, we presented a simulated-annealing-based graph drawing algorithm, taking the preservation of the mental map into account. The mental map criteria help us better preserve the contour and relative positions of the redrawn graph when a slight modification is made to the original graph, which in turn allowing the user to spend less time to relearn the new graph. Preserving the mental map and optimizing other aesthetic constraints are often in conflict with one another. Our implementation also allows the user to adjust the weights of different criteria.

In addition to the factors considered in this paper, graph features such as colour, node size, shape or line width may also play a key role in the preservation of the mental map. See (Franconeri et al., 2005) and (Rensink, 2002), for example, for work in this area from a psychological viewpoint. It is of interest and importance to take such additional features of graphs into account in the design of mental-map preserving graph drawing algorithms.

6 References

Böhringer, K. and Paulisch, F. (1990): Using Constraints to Achieve Stability in Automatic Graph Layout Algorithms. *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 43-51.

Bridgeman, S. and Tamassia R. (2002): A User Study in Similarity Measures for Graph Drawing. *Journal of Graph Algorithms and Applications*, **6**(3): 225-254.

Davidson, R. and Harel D. (1996): Drawing Graphs Nicely Using Simulated Annealing. *ACM Transactions on Graphics*, **15**(4):301-331.

Eades, P. (1984): A heuristic for graph drawing. *Congressus Numerantium*, **42**:149–160.

Eades, P., Lai, W., Misue, K., and Sugiyama K. (1991): Preserving the Mental Map of a Diagram. *Proc. of Compugraphics*, **91**: 24-33.

Fernandez-Duque, D. and Thornton, I. (2000): Change Detection Without Awareness: Do Explicit Report Underestimate the Representation of Change in the Visual System? *Visual Cognition*, **7**: 323-344.

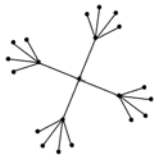
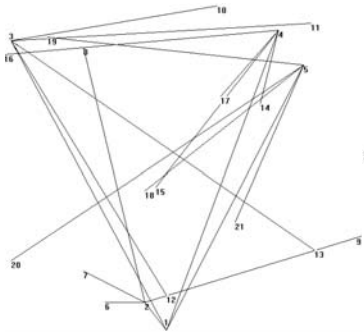
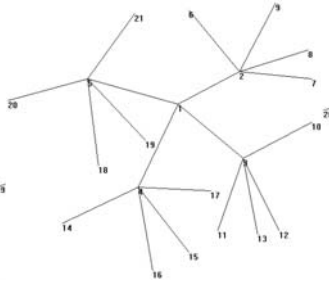


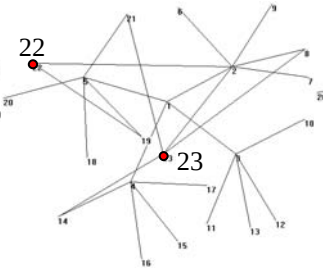
Figure 7. A tree example.



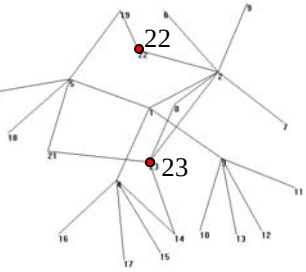
(a) the initial drawing.



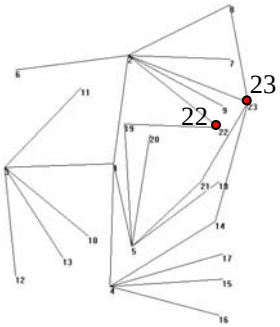
(b) applying "nice criteria" to (a).



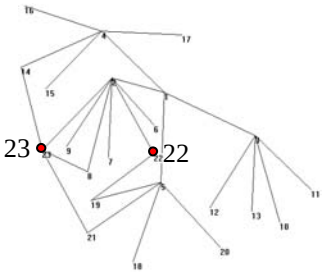
(c) adding nodes 22 and 23.



(d) applying "mental map criteria" to (c).

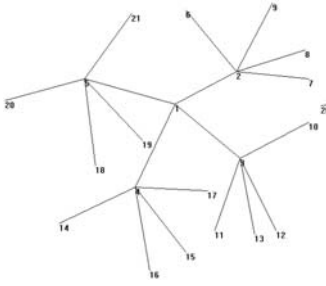


(e) applying "nice criteria" to (c).

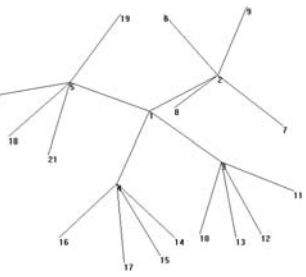


(f) applying "nice criteria" to a random drawing of the graph of (c).

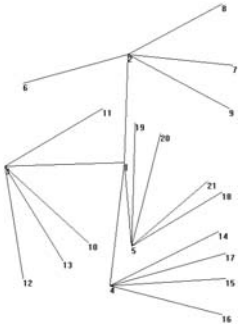
Figure 8. Various drawings illustrating the preservation of the mental map.



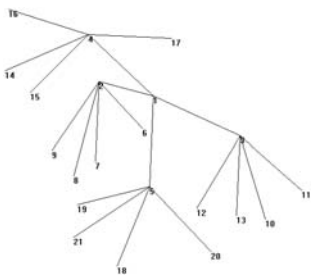
(a) Figure 8(c) without nodes 22 and 23.



(b) Figure 8(d) without nodes 22 and 23.

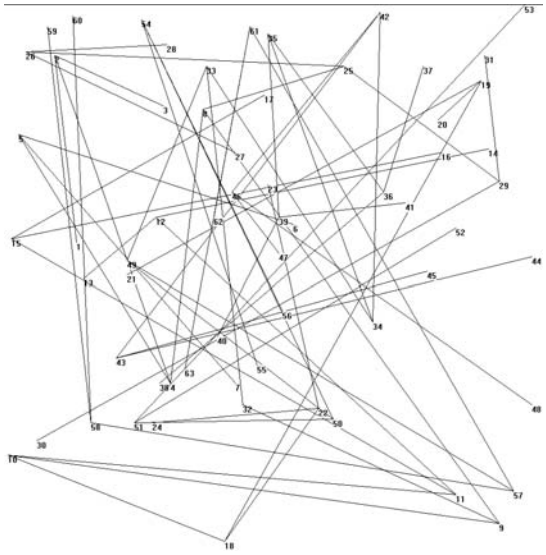


(c) Figure 8(e) without nodes 22 and 23.

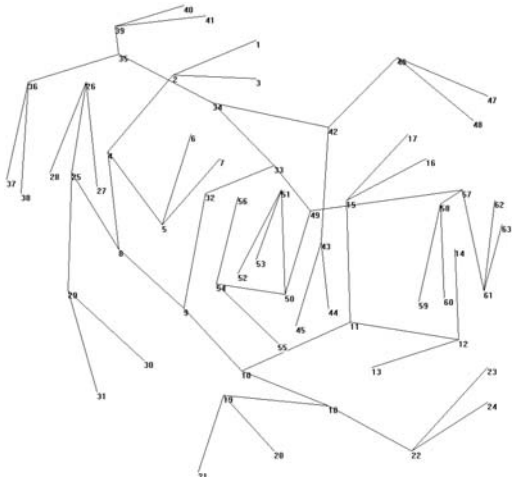


(d) Figure 8(f) without nodes22 and 23.

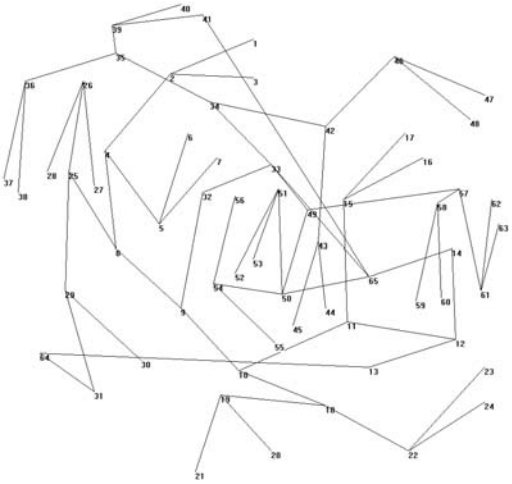
Figure 9. Drawings obtained from Figure 8 by deleting nodes 22 and 23.



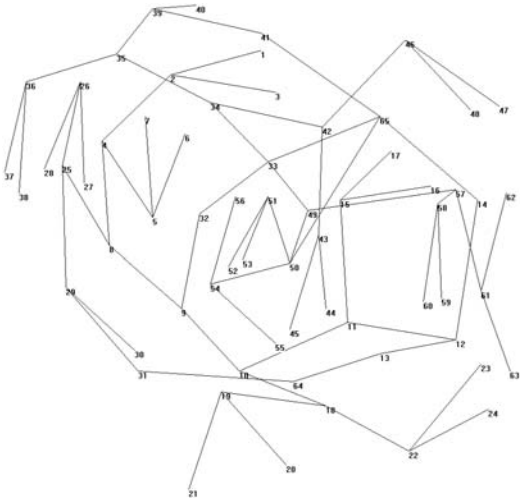
(a) the initial drawing.



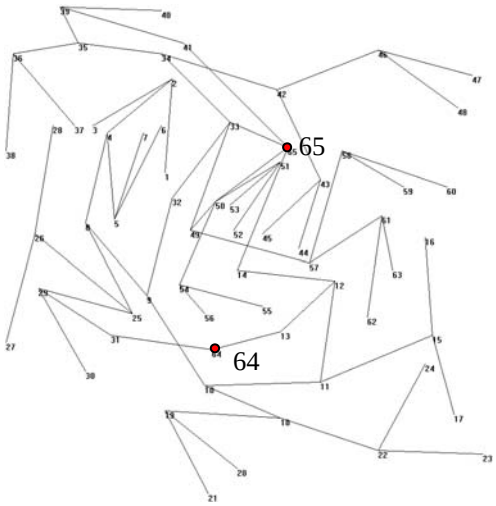
(b) applying “nice criteria” to (a).



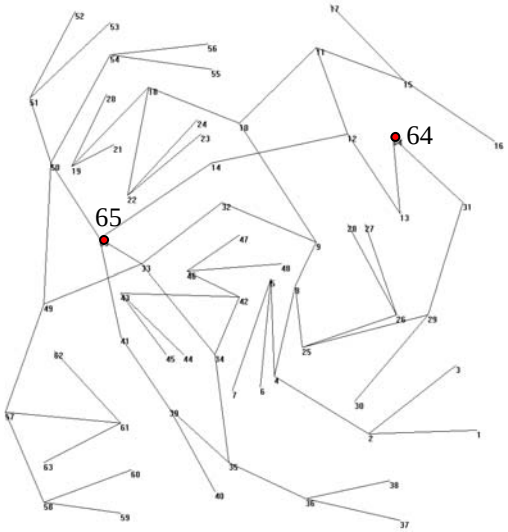
(c) adding nodes 64 and 65.



(d) applying “mental map criteria” to (c).

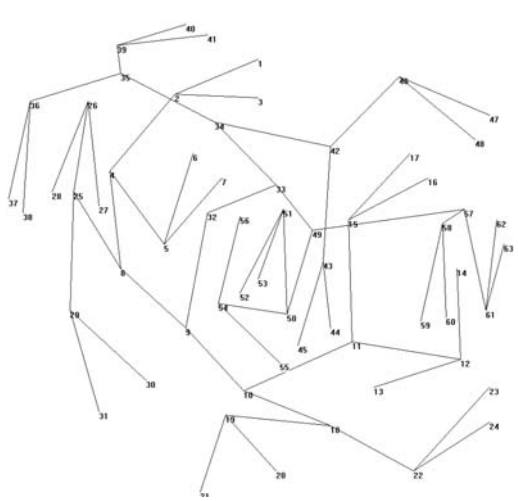


(e) applying “nice criteria” to (c).

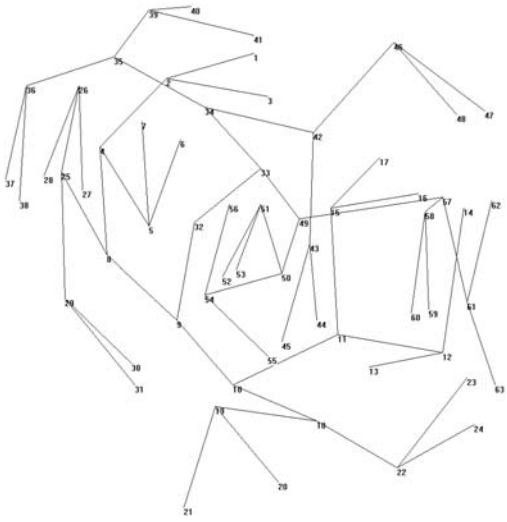


(f) applying “nice criteria” to a random drawing of the graph of (c).

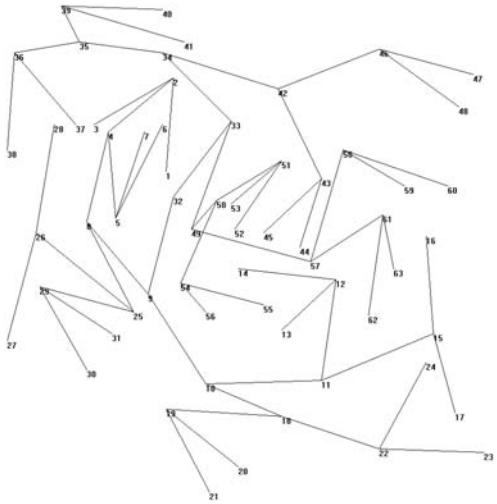
Figure 10. A larger tree example.



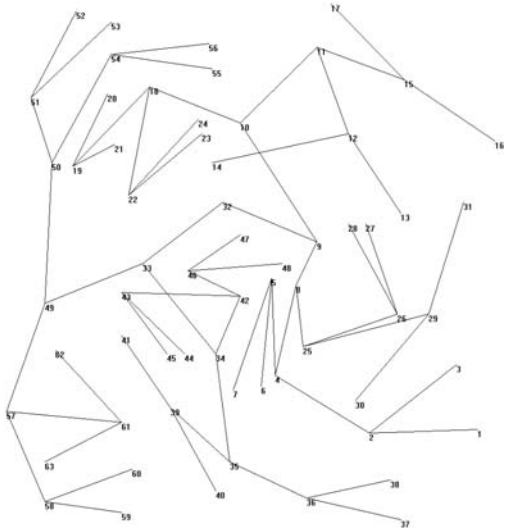
(a) Figure 10(c) without nodes 64 and 65.



(b) Figure 10(d) without nodes 64 and 65.



(c) Figure 10(e) without nodes 64 and 65.



(d) Figure 10(f) without nodes 64 and 65.

Figure 11. Drawings obtained from Figure 10 by deleting nodes 64 and 65.

Franconeri, S., Hollingworth, A., and Simons, D. (2005): Do New Objects Capture Attention? *Psychological Science*, **16**(4): 275-281.

Harel, D. and Sardas, M. (1998): Randomized Graph Drawing with Heavy-Duty Preprocessing. *J. Visual Language and Computing*, **6**: 233-253.

Harrison, J., Rensink, R., and van de Panne, M. (2004): Obscuring Length Changes During Animated Motion. *ACM Transactions on Graphics*, **23**: 569-573.

He, W. and Marriott, K. (1998): Constrained Graph Layout. *Constraints*, **3**(4): 289-314.

Kaufmann, M. and Wiese, R. (2002): Maintaining the Mental Map for Circular Drawings. *Proc. of Graph Drawing GD'02*, volume 2528 of *Lecture Notes in Computer Science*, 12-22.

Misue, K., Eades, P., Lai, W., and Sugiyama, K. (1995): Layout Adjustment and the Mental Map. *J. Visual Languages and Computing*, **6**:183-210.

Paulisch, F. and Tichy, W. (1990): Edge: an Extendible Directed Graph Editor. *Software – Practice and Experience*, **20**(S1):63-88.

Rensink, R. (2002): Change Detection. *Annual Review of Psychology*, **53**: 245-277.

Storey, M. and Müller, H. (1995): Graph Layout Adjustment Strategies. *Proc. of Graph Drawing GD'95*, volume 1207 of *Lecture Notes in Computer Science*, 487-99.

van Laarhoven, P. and Aarts, E. (1987): Simulated Annealing: Theory and Applications. Dordrect: Reidel.