BMC Bioinformatics

Research article

Open Access

An efficient grid layout algorithm for biological networks utilizing various biological attributes

Kaname Kojima†, Masao Nagasaki*†, Euna Jeong, Mitsuru Kato and Satoru Miyano

Address: Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan

Email: Kaname Kojima - kaname@ims.u-tokyo.ac.jp; Masao Nagasaki* - masao@ims.u-tokyo.ac.jp; Euna Jeong - eajeong@ims.u-tokyo.ac.jp; Mitsuru Kato - mitsuru@ims.u-tokyo.ac.jp; Satoru Miyano - miyano@ims.u-tokyo.ac.jp

* Corresponding author †Equal contributors

Published: 6 March 2007

BMC Bioinformatics 2007, **8**:76 doi:10.1186/1471-2105-8-76

Received: 23 October 2006
Accepted: 6 March 2007

This article is available from: http://www.biomedcentral.com/1471-2105/8/76

Abstract

Background: Clearly visualized biopathways provide a great help in understanding biological systems. However, manual drawing of large-scale biopathways is time consuming. We proposed a grid layout algorithm that can handle gene-regulatory networks and signal transduction pathways by considering edge-edge crossing, node-edge crossing, distance measure between nodes, and subcellular localization information from Gene Ontology. Consequently, the layout algorithm succeeded in drastically reducing these crossings in the apoptosis model. However, for larger-scale networks, we encountered three problems: (i) the initial layout is often very far from any local optimum because nodes are initially placed at random, (ii) from a biological viewpoint, human layouts still exceed automatic layouts in understanding because except subcellular localization, it does not fully utilize biological information of pathways, and (iii) it employs a local search strategy in which the neighborhood is obtained by moving one node at each step, and automatic layouts suggest that simultaneous movements of multiple nodes are necessary for better layouts, while such extension may face worsening the time complexity.

Results: We propose a new grid layout algorithm. To address problem (i), we devised a new force-directed algorithm whose output is suitable as the initial layout. For (ii), we considered that an appropriate alignment of nodes having the same biological attribute is one of the most important factors of the comprehension, and we defined a new score function that gives an advantage to such configurations. For solving problem (iii), we developed a search strategy that considers swapping nodes as well as moving a node, while keeping the order of the time complexity. Though a naïve implementation increases by one order, the time complexity, we solved this difficulty by devising a method that caches differences between scores of a layout and its possible updates.

Conclusion: Layouts of the new grid layout algorithm are compared with that of the previous algorithm and human layout in an endothelial cell model, three times as large as the apoptosis model. The total cost of the result from the new grid layout algorithm is similar to that of the human layout. In addition, its convergence time is drastically reduced (40% reduction).

Background

Modeling and simulations of large scale biological pathways are some of the most important tasks in Bioinformatics. Many applications, e.g., Cell Illustrator [1,2], Cytoscape [3], Pajek [4], PATIKA [5,6], and CADLIVE [7,8] have been developed in this area. Related to these topics, the visualization of biopathways is considered to play a key role in understanding biological systems. However, manual drawing of large-scale biopathways is a time consuming work, hence suitable biopathway layout algorithms and their applications are strongly demanded.

Biopathways are categorized into three types, i.e., metabolic pathways, signal transduction pathways, and gene-regulatory networks. For metabolic pathways, several algorithms have been already proposed [9-13], and some of them succeeded in capturing the flow of the reactions well. In contrast, few layout algorithms that provide a convenient biological understanding have been proposed for signal transduction pathways [14,15] and gene-regulatory networks [16,17]. Thus, our new layout algorithm is focused on signal transduction pathways and gene-regulatory networks. For signal transduction pathways and gene-regulatory networks, extant layout algorithms can be categorized into two types; force-directed and grid layout algorithms.

Force-directed algorithms are used in [16,17] by taking into account the directional constraint following different types of molecular and simple regional constraints from subcellular localizations. These algorithms have been successfully integrated into PATIKA. However, as pointed out in [14], force-directed algorithms may not be suitable for compact layouts of complex biopathways. Furthermore, intricately shaped regions such as torus-shaped region cannot be handled well as regional constraints in these force-directed algorithms. Hence, they are not suitable for models containing torus-shaped plasma membrane and nuclear membrane although such types of models are common as biopathways.

A grid layout algorithm (referred to as LK-grid layout algorithm) was initially proposed by Li and Kurata. The grid layout algorithm restricts the positions of all nodes to grid points. Li and Kurata defined a cost function for two nodes that depends on some distance between these nodes and the topology of their connections in the graph. They applied LK-grid layout algorithm to a yeast cell-cycle pathway and concluded that this algorithm can geometrically classify the pathway into functional categories without using biological information. Moreover, they noticed that the algorithm generates compact layouts while avoiding overlaps between nodes. [15] proposed CB-grid layout algorithm, in which so as to reduce edge-edge crossings and node-edge crossings, a penalty for these cases is added to the cost function. The algorithm can also deal with any complex regional constraints following subcellular localizations, and besides search space is reduced due to these constrains. As a result, in the apoptosis model, the layout algorithm succeeded in a drastic reduction of edge-edge crossings and node-edge crossings, while placing nodes in biologically proper regions.

However, in the case of larger-scale networks, this algorithm encountered three problems. First, a layout with randomly placed nodes is used as the initial layout. This random layout contains a large number of edge-edge crossings and node-edge crossings; subsequently, many iterations will be required to obtain a locally optimal layout. Secondly, although one of the features of CB-grid layout algorithm is to use the subcellular localization information, it still does not fully utilize biological characteristics. For example, it does not consider such biological attributes as types of entities (protein, mRNA, and microRNA) or types of processes (phosphorylation, binding, and translation), although in human layouts these biological attributes are apt to contribute to the comprehension of interesting biopathways easier. Thirdly, according to a greedy strategy, CB-grid layout algorithm updates a layout by moving one node at each step until the layout reaches an optimum. However, resulting layouts are just local optima, hence their quality fundamentally depends on the initial layout. Although in [15] a multi-step CB-grid layout algorithm was also proposed to solve this drawback, it requires higher time complexity and hence is not suitable for practical applications.

To overcome these three problems, we propose a new grid layout algorithm. For the first problem, we propose a new force-directed algorithm whose output is suitable as the initial layout of grid layout algorithms. For the second problem, we introduce the concept that assigns a score i.e., a negative cost, to a layout depending on how nodes with the same attribute are aligned. This concept is realized with a combo score function, which is combined with the cost function defined in CB-grid layout algorithm. For the third problem, the search strategy in CB-grid layout algorithm is improved by adding the swap operation while keeping the time complexity. By the swap operation, the new grid layout can also consider layouts generated by exchanging the positions of two nodes in the current layout at each step.

The Methods section is organized as follows: (i) first, we introduce the previous grid layout, i.e., CB-grid layout algorithm; (ii) for the first improvement in the initial layout of CB-grid layout algorithm, the new force-directed algorithm termed Eades initial layout algorithm is described; (iii) for the second improvement, CCB-grid layout algorithm, which is CB-grid layout algorithm with the combo score function is described; (iv) for the third improvement, SCCB-grid layout algorithm, which enhances CCB-grid layout algorithm by adding the swap operation is presented. In the Results and Discussion section, the performances of these new algorithms are compared and verified by applying them to the signal transduction pathway of an endothelial cell, which is larger than the pathways in [14] and [15].

Methods

CB-grid layout algorithm: Introduction of the grid layout algorithm

Given a graph $G = (V, E)$ with nodes V and edges E, a *layout* $L = (V, E, U, P)$ of G consists of the underlying graph G, grid points U and a function $P : V \rightarrow U$ such that $P(v_\alpha) \neq P(v_\beta)$ for any two distinct nodes $v_\alpha, v_\beta \in V$. This definition does not allow overlaps between nodes in the layout. For a layout L, this paper uses the following notations.

- W_L: a set of vacant points of L.
- E_v: the set of all edges connected to node v.
- $|V|$: the number of nodes in V.
- $|W|$: the number of vacant points in L, instead of $|W_L|$ if there is no confusion possible.

We define the following operations.

- $T_{v \rightarrow p} L$: the layout generated by moving a node v to a vacant point $p \in W_L$.
- $S_{v_\alpha \leftrightarrow v_\beta} L$: the layout generated by swapping nodes v_α and v_β.
- $D_v L$: the layout generated by removing a node v and all edges connected to v.

In addition, we define the following functions.

- $Cross_{e_i,e_j}(L)$: a binary function that returns 1 if an edge e_i crosses with an edge e_j and 0 otherwise.
- $Cross_{v_i,e_j}(L)$: a binary function that returns 1 if an edge e_j crosses with a node v_i and 0 otherwise.
- $Distance_{v_i,v_j}(L)$: a function that returns $w_{v_i,v_j} \cdot md(v_i, v_j)$, where w_{v_i,v_j} is the weight to the couple of nodes v_i and v_j, and $md(v_i, v_j)$ is the Manhattan distance between v_i and v_j.

In our previous approach [15] (mainly referred to as CB-grid layout algorithm), the *layout cost* $C(L)$ of L was defined as follows:

$$C(L) = W_{ee} \sum_{e_i,e_j \in E} Cross_{e_i,e_j}(L) + W_{ne} \sum_{v_k \in V, e_l \in E} Cross_{v_k,e_l}(L) + W_{dc} \sum_{v_m,v_n \in V} Distance_{v_m,v_n}(L), \quad (1)$$

where W_{ee}, W_{ne}, and W_d are called respectively *edge-edge crossing weight*, *node-edge crossing weight*, and *distance cost weight*.

The CB-grid layout algorithm repeats the operation of moving a unique node to a vacant point one-by-one until it reaches a locally optimal layout. At each step, the algorithm calculates costs of all layouts that can be generated by moving one of all nodes to one of all vacant points. The layout with the lowest cost is selected as a starting layout for the next step. After reaching convergence, the algorithm outputs a locally optimal layout. If the cost calculation of all possible adjacent layouts is implemented in a naïve way, high time complexity is required. To overcome this problem, the previous method [15] introduced Δ matrix that stores each possible cost difference at the previous step and succeeded in reducing the time complexity at each step from $O(|W|(|V|^2 + |E|^2)$ to $O(|V|^2 + |E|^2 + |W||E_{v_\beta}|(|V| + |E|))$, where v_β is the node moved at the previous step.

When CB-grid layout algorithm was applied to several biopathways, we encountered three problems. Thus, we propose new grid layout algorithms that solve these problems. Problems and solutions are summarized as follows:

1. Improving the choice of the initial layout: since a locally optimal layout depends noticeably on the initial layout, we first apply Eades initial layout algorithm to a random layout, and use its output as the initial layout. In the previous approach, a random layout was directly used as the initial layout.

2. Improving the cost function: we introduce the concept of a combo score that gives a good score, i.e., a negative cost when nodes with the same biological attribute are aligned (CCB-grid layout algorithm). In CB-grid layout algorithm, the biological attributes, except subcellular localization, were ignored.

3. Improving the search strategy: we propose a better search strategy, which allows us to obtain improved results, keeping the time complexity. For obtaining a bet-

ter layout, the search space is extended by adding the swap operation. At each step, all layouts obtained by swapping two nodes are also considered (SCCB-grid layout algorithm).

In the remainder of this section, we describe these three new algorithms mentioned above.

Eades initial layout algorithm: generating a new initial layout for grid layout algorithms

In the previous paper [15], a random layout was used as an initial layout for CB-grid layout algorithm. When the initial layout is far from the global optimum, the local optimum obtained tends to be unacceptable. Therefore, we decided to develop Eades algorithm [18] and use its output as the initial layout. Eades algorithm is one of the force-directed algorithms, consisting of the following two steps.

1. Two types of forces are defined for each pair of nodes. If two nodes are adjacent, there exists an attractive force $a_{c1}\log(d/a_{c2})$ between them, where a_{c1} and a_{c2} are constants, and d is the distance between the two nodes. On the other hand, if two nodes are not adjacent, there exists a repulsive force $r_c/\sqrt{d}$ between them, where r_c is a constant. At each step, the positions of all the nodes are updated according to the sum of the repulsive and attractive forces between them.

2. The above step is iterated a predetermined number of times, and the final result is obtained.

We have customized two points in Eades algorithm. First, nodes in Eades algorithm can be placed anywhere. All the nodes in the initial layout for CB-grid layout algorithm, however, should be placed on the grid points that satisfy the subcellular localization. Thus, the output of Eades algorithm cannot be used directly as an input for CB-grid layout algorithm.

To handle this problem, we propose to move each node to the closest vacant point that satisfies the subcellular localization after moving nodes at each step.

Second improvement is the following one. Since Eades algorithm doesn't consider edge-edge crossings and node-edge crossings in its implementation, the resulting layout could contain a lot of such crossings. For example, suppose a biological pathway with a subcellular localization, membrane, which slimly surrounds other subcellular localizations as shown in Figure 1(a), the graph in (a) could be a layout resulting from Eades algorithm. In this case, the layout might contain a large number of edge-edge crossings and node-edge crossings because edges cross over other subcellular localizations. In order to avoid this problem, we propose to gather nodes around a particular grid point for each subcellular localization as shown in Figure 1(b). Eades algorithm with the above improvements is called *Eades initial layout algorithm*.

CCB-grid layout algorithm: utilizing various biological attributes

When humans draw biopathway models, nodes with the same attribute are usually arranged according to a rule. In CB-grid layout algorithm, this type of information is completely ignored. To implement this type of property, we introduce the concept of combo scores called **combo1** and **combo2** (see Figure 2). Note that a combo score is applied only to nodes having an attribute since some nodes do not have any attributes. We denote the set of nodes having an attribute by $V' \subseteq V$. In this algorithm, (i) upperGrid(p, i)/lowerGrid(p, i) returns the upper/lower ith grid point over/under a grid point $p \in P$, and (ii) Attr(v) is the attribute of a node $v \in V'$, and $CW_a = (1 + C/|V'_a|)$, where C is a constant and normally set to $|V|$, and V'_a is the set of nodes having an attribute a.

The combo score is designed such that the more nodes with the same attribute are aligned vertically, the higher the score is. The combo score is defined between two nodes, and a combo score of a layout L is defined to be the sum of all the combo scores occurring in L. We say that two nodes have a *combo relation* when a combo score occurs between them. Note that the horizontal alignment score is not implemented because if the above combo score supported both the vertical and horizontal directions, the numbers of edge-edge crossings and node-edge crossings would be considerably increased. Therefore, we should choose only one direction for combo scores. In this paper, we defined combo scores in the vertical direction. We have considered two types of combo scores, i.e., **combo1** and **combo2** for layouts in Figure 3(a) and 3(b), respectively. Let nodes v_a to v_f in Figure 3 have the same attribute. The **combo1** considers only the nodes with one vertical grid distance from the target node. In contrast, **combo2** considers the nodes with up to two vertical grid distances from the target node. For the layout in Figure 3(a), the number of combo relations with **combo1** and **combo2** are 8 and 12, respectively. If node v_f is moved as shown in Figure 3(b), the number of combo relations with **combo1** is the same as before, whereas that with **combo2** is 14. Thus, only by using **combo2**, we can improve the combo score when node v_f is moved as

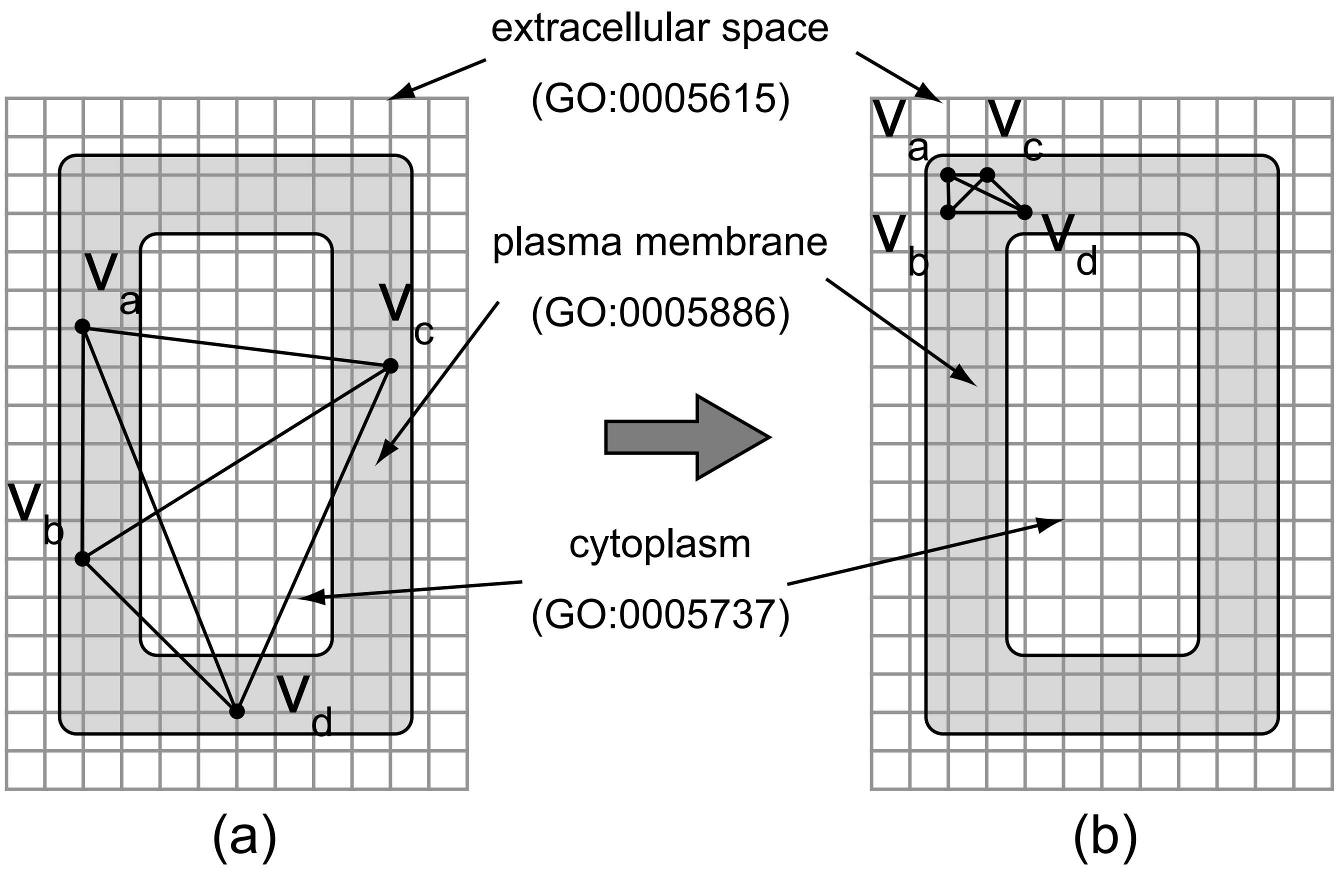

Figure 1
Two layouts with the same canvas and three subcellular localizations. The grid canvases (a) and (b) have the same biological subcellular localizations extracellular space, plasma membrane, and cytoplasm. Both canvases contain the same graph with four nodes that are located in plasma membrane, which surrounds cytoplasm. In (a), nodes are spread apart in plasma membrane, and edges among these nodes cross over cytoplasm. In (b), the nodes are gathered in the left-top corner, and no edge crosses over cytoplasm. Due to its crossing patterns in (a) these edges have a higher probability to cross other nodes in cytoplasm. This is the drawback of using the layout in (a) as the initial layout for Eades initial layout algorithm.

shown in Figure 3(a) and 3(b). As shown in the dotted rectangle in Figure 3(a), a pair of vertically aligned nodes often occurs during the process of updating a layout. In this case, Figure 3(b) should be a better layout than Figure 3(a). For this reason, we decide to employ **combo2**. Henceforth, for a node $v \in V$ in a layout L, $Combo_v$ (L) denotes the same combo score as **combo2** (v, L). The total score $\sum_{v \in V} Combo_v(L)$ for L is denoted by $Combo$ (L).

If CW_a returns the same value for any attribute a, many of the nodes with the same attribute will be vertically aligned easily since they have a greater chance to neighbor one another. So as to reduce the biases among the attributes, we define CW_a to be inversely related to the total number of the nodes whose attribute is a.

By modifying the layout score of CB-grid layout algorithm, we can define the layout cost C (L) of a layout L with the new concept of the combo score as follows:

$$\begin{aligned} C(L) &= W_{ee} \sum_{e_i, e_j \in E} Cross_{e_i, e_j}(L) + W_{ne} \sum_{v_k \in V, e_l \in E} Cross_{v_k, e_l}(L) + W_{dc} \sum_{v_m, v_n \in V} Distance_{v_m, v_n}(L) \\ &- W_{cs}\left(\frac{1}{2} \sum_{v_o \in V'} Combo_{v_o}(L) \right), \end{aligned} \tag{2}$$

where W_{cs} is called *combo score weight*. CB-grid layout algorithm improved by the above modification is named *Combo score, Cross cost and Biological information grid layout algorithm* (CCB-grid layout algorithm). The reason for multiplying the sum of the combo scores by 1/2 is that combo scores are counted twice since a combo score between nodes v_α and v_β is included in both $Combo_{v_\alpha}$ (L)

```
     combo1($v \in V', L$)
 1:  $numCombo \leftarrow 0$
 2:  $p' \leftarrow P(v)$
 3:  $p'' \leftarrow$ upperGrid($p'$, 1)
 4:  if isCombo($v, p''$) then
 5:     $numCombo \leftarrow numCombo + 1$
 6:  end if
 7:  $p' \leftarrow p(v)$
 8:  $p'' \leftarrow$ lowerGrid($p'$, 1)
 9:  if isCombo($v, p''$) then
10:     $numCombo \leftarrow numCombo + 1$
11:  end if
12:  return $CW_{\mathrm{Attr}(v)} \cdot numCombo$
```

```
     combo2($v \in V', L$)
 1:  $numCombo \leftarrow 0$
 2:  $p' \leftarrow P(v)$
 3:  $p'' \leftarrow$ upperGrid($p'$, 1)
 4:  if isCombo($v, p''$) then
 5:     $numCombo \leftarrow numCombo + 1$
 6:     $p''' \leftarrow$ upperGrid($p''$, 1)
 7:     if isCombo($v, p'''$) then
 8:        $numCombo \leftarrow numCombo + 1$
 9:     end if
10:  end if
11:  $p' \leftarrow p(v)$
12:  $p'' \leftarrow$ lowerGrid($p'$, 1)
13:  if isCombo($v, p''$) then
14:     $numCombo \leftarrow numCombo + 1$
15:     $P''' \leftarrow$ lowerGrid($p''$, 1)
16:     if isCombo($v, p'''$) then
17:        $numCombo \leftarrow numCombo + 1$
18:     end if
19:  end if
20:  return $CW_{\mathrm{Attr}(v)} \cdot numCombo$
```

```
     isCombo($v_\alpha \in V', p \in U$)
 1:  if there exists a node $v_\beta \in V'$ on $p$ then
 2:     if Attr($v_\alpha$) and Attr($v_\beta$) are the same then
 3:        return true
 4:     end if
 5:  end if
 6:  return false
```

Figure 2
Pseudo codes of combo score functions: combo1 and combo2. (a) **combo1**: a score function that considers nodes with one vertical grid distance from the target node. (b) **combo2**: a score function that considers nodes with up to two vertical grid distances from the target node, (c) isCombo: a boolean function that takes a node and a grid point as its arguments and returns "true" if the attribute of the node and that of the node on the grid point are the same.

and $Combo_{v_\beta}$ (L). The algorithm is the same as *C-optimization* (L) step in [15] except for the use of the above layout cost $C(L)$, i.e., the algorithm for calculating Δ matrix is also the same.

For calculating the combo score for each node, only four nodes need to be checked at most, i.e., its time complexity is constant, while for calculating the edge-edge crossing cost, the node-edge crossing cost, and the distance cost for each node, these time complexities depend on $|E|$, $|V|$, and $|W|$, respectively. Thus, without using Δ matrix, the time complexity related to combo scores is $O(|V||W|)$ at each step.

At each step, we need to calculate the difference between the combo score of the previous layout L and that of the current layout that is generated by moving a node v to a vacant point p, i.e., $Combo(T_{v\rightarrow p}L) - Combo(L)$. We can efficiently calculate the difference of the combo score Δ^{cs}_{vp} (L) as follows:

$$\Delta^{cs}_{vp}(L) = \begin{cases} W_{cs}(Combo_v(T_{v\rightarrow p}L) - Combo_v(L) + Adj_v(T_{v\rightarrow p}L) - Adj_v(L)) & \text{if } v \in V' \\ 0 & \text{if } v \notin V' \end{cases} \qquad (3)$$

where

$$Adj_v = \begin{cases} CW_{\mathrm{Attr}(v)} & \text{if } \begin{array}{l} \text{isCombo}(v, \text{upperGrid}(\text{P}(v,1))) = \textbf{true} \;\& \\ \text{isCombo}(v, \text{lowerGrid}(\text{P}(v,1))) = \textbf{true} \end{array} \\ 0 & \text{otherwise.} \end{cases} \qquad (4)$$

We introduced $Adj_v(L)$ due to the following reason. First, suppose that three nodes with the same attribute are aligned vertically. We call them v_α, v_β, and v_γ beginning from the bottom. There are three combo relations among the three nodes: one is between v_α and v_β, another

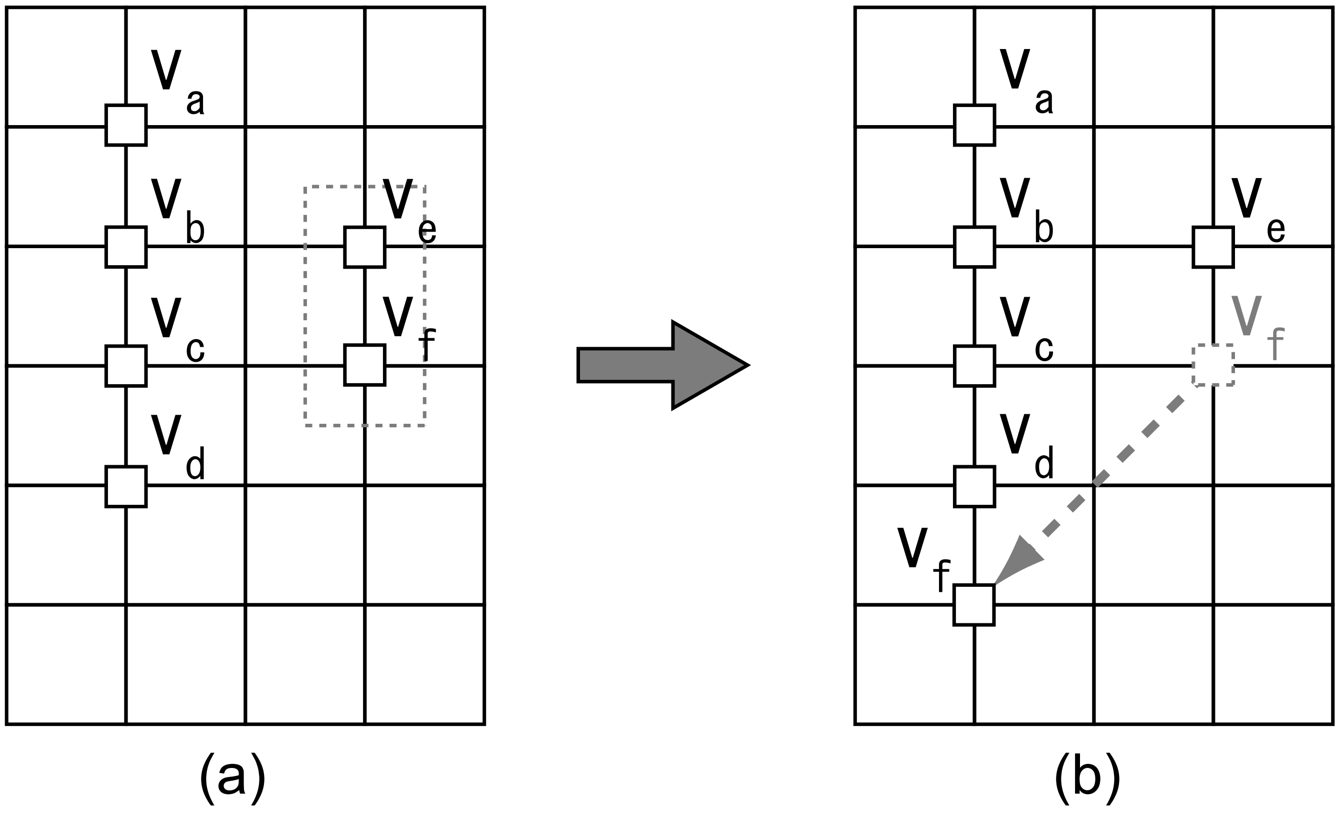

Figure 3
An example that compares the features of combo1 and combo2 score functions. (a) An intermediate layout of CCB-grid layout algorithm. In this layout, all six nodes have the same attribute. (b) The next candidate layout that is generated from (a) by moving node v_f below node v_d. Combo scores of (a) and (b) are the same with **combo1** score function. Instead, the combo score of (b) will be better than (a) with **combo2** score function.

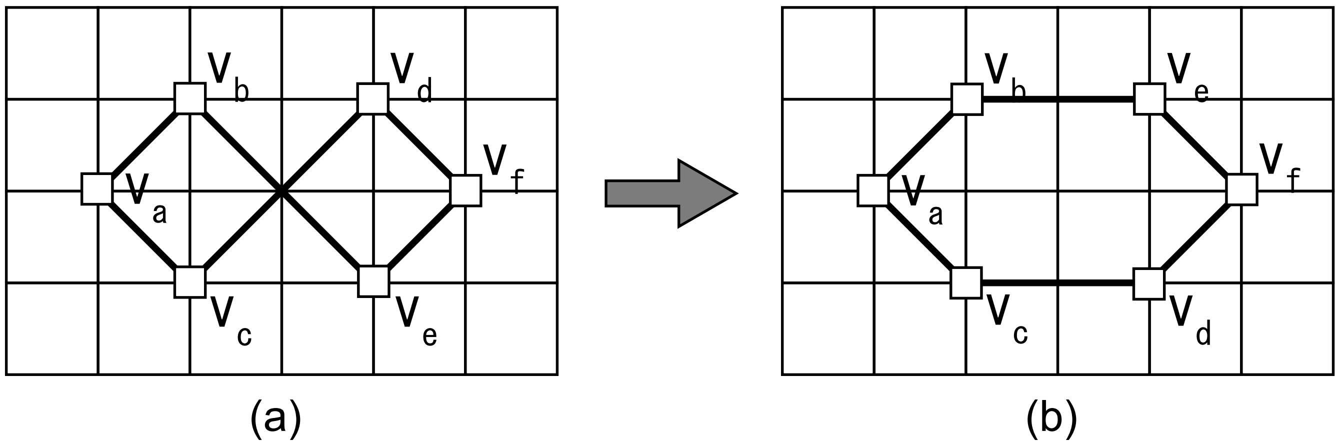

Figure 4
An optimal layout of CB-grid and improved layout with the swap operation. (a) An optimal layout for CB-grid layout algorithm. (b) From (a) a better layout will be generated with the swap operation.

between v_β and v_γ and the third between v_α and v_γ Although $v_\beta$is involved in these three combo relations, the combo relation between v_α and $v_\gamma$is not considered in $Combo_{v_\beta}$ (L). Therefore, $Adj_v(L)$ is needed to correct this type of undercount.

SCCB-grid layout algorithm: extension of the search space due to the swap operation

Another drawback of CB-grid layout algorithm is that only one node can be moved to a vacant point at each step. For example, the layout shown in Figure 4(a) is optimal for CB-grid layout algorithm despite the fact the layout in Figure 4(b) should be selected as the better layout. This limitation is due to the strategy of CB-grid layout algorithm. Thus, we have devised a new algorithm by allowing the swap operations between two nodes while keeping the time complexity. With this improvement, the layout in Figure 4(a) will be arranged as shown in Figure 4(b). The new algorithm is named CCB-grid layout with the swap operation (SCCB-grid layout algorithm). The layout cost function is the same as in CCB-grid layout algorithm. However, a naïve implementation would increase the time complexity to calculate the layout cost for swapped layouts.

In the previous approach [15], Δ matrix stores cost differences that are induced only by moving nodes to vacant points. As a result, if a grid point of interest was occupied at the previous step, we cannot exploit Δ matrix to calculate cost differences corresponding to that grid point. Since grid points of interest on the swap operation are obviously occupied at the previous step, Δ matrix cannot be used. However, if Δ matrix also stores cost differences related to occupied points, Δ matrix can be exploited for this problematic case, too. We then propose an extended Δ matrix, which considers occupied points as well as vacant points. Since the definition of the cost differences for vacant points cannot be applied directly to occupied points, we decide to calculate the cost differences for the occupied points by calculating it without taking into account the node occupying that grid point and all edges connected to it. In the remainder of this section, we will show how to calculate the extended Δ matrix and then compare the time complexity of the extended Δ matrix and the original Δ matrix.

Henceforth, let us refer to the extended Δ matrix as Δ matrix. Given a layout L, at the first step, we update Δ (L) matrix as follows:

$$\Delta_{v_\alpha p}(L) = \begin{cases} F_{v_\alpha}(T_{v_\alpha \to p}L) - F_{v_\alpha}(L) & \text{if } p \in W_L \\ F_{v_\alpha}(T_{v_\alpha \to p}D_{v_\gamma}L) - F_{v_\alpha}(D_{v_\gamma}L) & \text{if } p = P(v_\gamma). \end{cases} \qquad (5)$$

F_{v_α} is the following function:

$$F_{v_\alpha}(L) = W_{ee}\sum_{e_i \in E_{v_\alpha}, e_j \in E} Cross_{e_i,e_j}(L) + W_{ne}\sum_{e_k \in E} Cross_{v_\alpha,e_k}(L) + W_{dc}\sum_{v_l \in V} Distance_{v_\alpha,v_l}(L). \qquad (6)$$

If the previous layout is updated by moving node v_β to vacant point q, Δ ($T_{v_\alpha \to p}L$) can be updated efficiently by using Δ (L) as follows:

$$\Delta_{v_\alpha p}(T_{v_\beta \to q}L) = \begin{cases} \Delta_{v_\beta p}(L) - \Delta_{v_\beta q}(L), & \text{if } v_\alpha = v_\beta, p \in W_{T_{v_\beta \to q}L} & (\text{case } 1) \\ \Delta_{v_\alpha p}(L) + DIFF_0, & \text{if } v_\alpha \neq v_\beta, p \in W_{T_{v_\beta \to q}L} \setminus P(v_\beta) & (\text{case } 2) \\ \Delta_{v_\alpha p}(L) + DIFF_1, & \text{if } v_\alpha \neq v_\beta, p = P(v_\beta) & (\text{case } 3) \\ \Delta_{v_\beta p}(L) - \Delta_{v_\beta q}(L) + DIFF_2, & \text{if } v_\alpha = v_\beta, p = P(v_\gamma) & (\text{case } 4) \\ \Delta_{v_\alpha p}(L) + DIFF_3 & \text{if } v_\alpha \neq v_\beta, p = P(v_\gamma) & (\text{case } 5) \\ \Delta_{v_\alpha p}(L) + DIFF_4, & \text{if } v_\alpha \neq v_\beta, p = q & (\text{case } 6), \end{cases} \qquad (7)$$

where $DIFF_0$ to $DIFF_4$ are defined in the following way:

$$DIFF_0 = Q_{v_\alpha,v_\beta}(T_{v_\alpha \to p}T_{v_\beta \to q}L) - Q_{v_\alpha,v_\beta}(T_{v_\beta \to q}L) - Q_{v_\alpha,v_\beta}(T_{v_\alpha \to p}L) + Q_{v_\alpha,v_\beta}(L) \qquad (8)$$

$$DIFF_1 = Q_{v_\alpha,v_\beta}(T_{v_\alpha \to p}T_{v_\beta \to q}L) - Q_{v_\alpha,v_\beta}(T_{v_\beta \to q}L) \qquad (9)$$

$$DIFF_2 = Q_{v_\beta,v_\gamma}(T_{v_\beta \to q}L) - Q_{v_\beta,v_\gamma}(L) \qquad (10)$$

$$DIFF_3 = Q_{v_\alpha,v_\beta}(T_{v_\alpha \to p}T_{v_\beta \to q}D_{v_\gamma}L) - Q_{v_\alpha,v_\beta}(T_{v_\beta \to q}D_{v_\gamma}L) - Q_{v_\alpha,v_\beta}(T_{v_\alpha \to p}D_{v_\gamma}L) + Q_{v_\alpha,v_\beta}(D_{v_\gamma}L) \qquad (11)$$

$$DIFF_4 = -Q_{v_\alpha,v_\beta}(T_{v_\alpha \to p}L) + Q_{v_\alpha,v_\beta}(L), \qquad (12)$$

where Q shall be defined below.

If the previous layout is updated by swapping two nodes v_{β_1} and v_{β_2}, Δ ($S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}L$) is then updated efficiently by using Δ (L) as follows:

$$\Delta_{v_\alpha p}(S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}L) = \begin{cases} \Delta_{v_{\beta_1} p}(L) - \Delta_{v_{\beta_1} P(v_{\beta_2})}(L) + DIFF_5, & \text{if } v_\alpha = v_{\beta_1}, p \in W_L & (\text{case } 1) \\ \Delta_{v_\alpha p}(L) + DIFF_6, & \text{if } v_\alpha \neq v_{\beta_1}, p \in W_L \setminus \{P(v_{\beta_1}), P(v_{\beta_2})\} & (\text{case } 2) \\ \Delta_{v_{\beta_1} p}(L) - \Delta_{v_{\beta_1} P(v_{\beta_2})}(L) + DIFF_7, & \text{if } v_\alpha = v_{\beta_1}, p = P(v_\gamma) & (\text{case } 3) \\ \Delta_{v_\alpha p}(L) + DIFF_8, & \text{if } v_\alpha \neq v_{\beta_1}, v_\alpha \neq v_{\beta_2}, p = P(v_\gamma) & (\text{case } 4) \\ \Delta_{v_\alpha p}(L) + DIFF_9, & \text{if } v_\alpha \neq v_{\beta_1}, v_\alpha \neq v_{\beta_2}, p = P(v_{\beta_2}) & (\text{case } 5), \end{cases} \qquad (13)$$

where $DIFF_5$ to $DIFF_9$ are defined in the following way:

$$DIFF_5 = Q_{v_{\beta_1},v_{\beta_2}}(T_{v_{\beta_1} \to p}S_{v_{\beta_1} \leftrightarrow v_{\beta_1}}L) - Q_{v_{\beta_1},v_{\beta_2}}(S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}L) - Q_{v_{\beta_1},v_{\beta_2}}(T_{v_{\beta_1} \to p}L) + Q_{v_{\beta_1},v_{\beta_2}}(L) \qquad (14)$$

$$DIFF_6 = \hat{Q}_{v_\alpha,v_{\beta_1},v_{\beta_2}}(T_{v_\alpha \to p}S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}L) - \hat{Q}_{v_\alpha,v_{\beta_1},v_{\beta_2}}(S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}L) - \hat{Q}_{v_\alpha,v_{\beta_1},v_{\beta_2}}(T_{v_\alpha \to p}L) + \hat{Q}_{v_\alpha,v_{\beta_1},v_{\beta_2}}(L) \qquad (15)$$

$$\begin{aligned} DIFF_7 &= Q_{v_{\beta_1},v_{\beta_2}}(T_{v_{\beta_1} \to p}S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}D_{v_\gamma}L) - Q_{v_{\beta_1},v_{\beta_2}}(T_{v_{\beta_1} \to p}D_{v_\gamma}L) - Q_{v_{\beta_1},v_{\beta_2}}(S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}D_{v_\gamma}L) \\ &+ Q_{v_{\beta_1},v_{\beta_2}}(D_{v_\gamma}L) + Q_{v_{\beta_1},v_\gamma}(T_{v_{\beta_1} \to P(v_{\beta_2})}D_{v_{\beta_2}}L) - Q_{v_{\beta_1},v_\gamma}(D_{v_{\beta_2}}L) \end{aligned} \qquad (16)$$

$$\begin{aligned} DIFF_8 &= \hat{Q}_{v_\alpha,v_{\beta_1},v_{\beta_2}}(T_{v_\alpha \to p}S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}D_{v_\gamma}L) - \hat{Q}_{v_\alpha,v_{\beta_1},v_{\beta_2}}(S_{v_{\beta_1} \leftrightarrow v_{\beta_2}}D_{v_\gamma}L) - \hat{Q}_{v_\alpha,v_{\beta_1},v_{\beta_2}}(T_{v_\alpha \to p}D_{v_\gamma}L) \\ &+ \hat{Q}_{v_\alpha,v_{\beta_1},v_{\beta_2}}(D_{v_\gamma}L) \end{aligned} \qquad (17)$$

$$\begin{aligned} DIFF_9 &= Q_{v_\alpha,v_{\beta_2}}(T_{v_\alpha \to P(v_{\beta_2})}T_{v_{\beta_2} \to P(v_{\beta_1})}D_{v_{\beta_1}}L) - Q_{v_\alpha,v_{\beta_2}}(T_{v_{\beta_2} \to P(v_{\beta_1})}D_{v_{\beta_1}}L) \\ &- Q_{v_\alpha,v_{\beta_1}}(T_{v_\alpha \to P(v_{\beta_2})}D_{v_{\beta_2}}L) + Q_{v_\alpha,v_{\beta_1}}(D_{v_{\beta_2}}L). \end{aligned} \qquad (18)$$

The case of $v_\alpha = v_{\beta_2}$ is not considered in Equation (13) because equations of this case can be obtained by simply replacing v_{β_1} with v_{β_2} in case 1 and 3.

$Q_{v_a,v_b}(\cdot)$ and $\hat{Q}_{v_a,v_b,v_c}(\cdot)$ in $DIFF_0$ to $DIFF_9$ are partial cost functions depending on the two nodes v_a and v_b and the three nodes v_a, v_b, and v_c, respectively, they are the sums of the corresponding partial edge-edge crossing costs, node-edge crossing costs and distance costs as follows:

$$Q_{v_a,v_b}(L) = W_v Q^{dc}_{v_a,v_b}(L) + W_{ee} Q^{ee}_{v_a,v_b}(L) + W_{ve} Q^{ve}_{v_a,v_b}(L) \qquad (19)$$

$$\hat{Q}_{v_a,v_b,v_c}(L) = W_v \hat{Q}^{dc}_{v_a,v_b,v_c}(L) + W_{ee} \hat{Q}^{ee}_{v_a,v_b,v_c}(L) + W_{ve} \hat{Q}^{ve}_{v_a,v_b,v_c}(L), \qquad (20)$$

where $Q^{ee}_{v_a,v_b}(\cdot)$ and $\hat{Q}^{ee}_{v_a,v_b,v_c}(\cdot)$ are related to edge-edge crossings, while $Q^{ne}_{v_a,v_b}(\cdot)$ and $\hat{Q}^{ne}_{v_a,v_b,v_c}(\cdot)$ are related to node-edge crossings, and $Q^{dc}_{v_a,v_b}(\cdot)$ and $\hat{Q}^{dc}_{v_a,v_b,v_c}(\cdot)$ are related to the distance cost. The details are described as below.

(a) $Q^{ee}_{v_a,v_b}(\cdot)$ is a partial edge-edge crossing cost function of E_{v_a} and E_{v_b}, and is defined as follows:

$$Q^{ee}_{v_a,v_b}(L) = \begin{cases} \displaystyle\sum_{e_{v_a}\in E_{v_a}, e_{v_b}\in E_{v_b}} Cross_{e_{v_a},e_{v_b}}(L) & \text{if } (v_a,v_b) \notin E \\ \displaystyle\sum_{e_{v_a}\in E_{v_a}, e_{v_b}\in E_{v_b}} Cross_{e_{v_a},e_{v_b}}(L) + \sum_{e\in E} Cross_{e,(v_a,v_b)}(L) & \text{if } (v_a,v_b) \in E \end{cases}. \qquad (21)$$

Similarly, $\hat{Q}^{ee}_{v_a,v_b,v_c}(\cdot)$ is a partial edge-edge crossing cost function of E_{v_a}, E_{v_b}, and E_{v_c}, and is defined as follows:

$$\hat{Q}^{ee}_{v_a,v_b,v_c}(L) = \begin{cases} \left.\begin{array}{l}\displaystyle\sum_{e_{v_a}\in E_{v_a}, e_{v_b}\in E_{v_b}} Cross_{e_{v_a},e_{v_b}}(L) \\ + \displaystyle\sum_{e_{v_a}\in E_{v_a}, e_{v_c}\in E_{v_c}} Cross_{e_{v_a},e_{v_c}}(L)\end{array}\right. (=\hat{Q}^{ee}) & \text{if } \begin{array}{l}(v_a,v_b)\notin E,\\ (v_a,v_c)\notin E\end{array} \\ \hat{Q}^{ee} + \displaystyle\sum_{e\in E\setminus E_{v_c}} Cross_{e,(v_a,v_b)}(L) & \text{if } \begin{array}{l}(v_a,v_b)\in E,\\ (v_a,v_c)\notin E\end{array} \\ \hat{Q}^{ee} + \displaystyle\sum_{e\in E\setminus E_{v_b}} Cross_{e,(v_a,v_c)}(L) & \text{if } \begin{array}{l}(v_a,v_b)\notin E,\\ (v_a,v_c)\in E\end{array} \\ \hat{Q}^{ee} + \displaystyle\sum_{e\in E\setminus E_{v_c}} Cross_{e,(v_a,v_b)}(L) + \sum_{e\in E\setminus E_{v_b}} Cross_{e,(v_a,v_c)}(L) & \text{if } \begin{array}{l}(v_a,v_b)\in E,\\ (v_a,v_c)\in E\end{array} \end{cases}. \qquad (22)$$

(b) $Q^{ne}_{v_a,v_b}$ is a partial node-edge crossing cost function of v_a, v_b, E_{v_a}, and E_{v_b}, and is defined as follows:

$$Q^{ne}_{v_a,v_b}(L) = \begin{cases} \displaystyle\sum_{e_{v_a}\in E_{v_a}} Cross_{v_b,e_{v_a}}(L) + \sum_{e_{v_b}\in E_{v_b}} Cross_{v_a,e_{v_b}}(L) \quad (=Q^{ne}) & \text{if } (v_a,v_b)\notin E \\ Q^{ne} + \displaystyle\sum_{v\in V} Cross_{v,(v_a,v_b)}(L) & \text{if } (v_a,v_b)\in E \end{cases} \qquad (23)$$

Similarly, $\hat{Q}^{ne}_{v_a,v_b,v_c}(\cdot)$ is a partial node-edge crossing cost function of v_a, v_b, v_c, E_{v_a}, E_{v_b}, and E_{v_c}, and is defined as follows:

$$\hat{Q}^{ne}_{v_a,v_b,v_c}(L) = \begin{cases} \left.\begin{array}{l}\displaystyle\sum_{e_{v_a}\in E_{v_a}} Cross_{v_b,e_{v_a}}(L) + \sum_{e_{v_a}\in E_{v_a}} Cross_{v_c,e_{v_a}}(L) \\ + \displaystyle\sum_{e_{v_b}\in E_{v_b}} Cross_{v_a,e_{v_b}}(L) + \sum_{e_{v_c}\in E_{v_c}} Cross_{v_a,e_{v_c}}(L)\end{array}\right. (=\hat{Q}^{ne}) & \text{if } \begin{array}{l}(v_a,v_b)\notin E,\\ (v_a,v_c)\notin E\end{array} \\ \hat{Q}^{ne} + \displaystyle\sum_{v\in V\setminus v_c} Cross_{v,(v_a,v_b)}(L) & \text{if } \begin{array}{l}(v_a,v_b)\in E,\\ (v_a,v_c)\notin E\end{array} \\ \hat{Q}^{ne} + \displaystyle\sum_{v\in V\setminus v_b} Cross_{v,(v_a,v_c)}(L) & \text{if } \begin{array}{l}(v_a,v_b)\notin E,\\ (v_a,v_c)\in E\end{array} \\ \hat{Q}^{ne} + \displaystyle\sum_{v\in V\setminus v_c} Cross_{v,(v_a,v_b)}(L) + \sum_{v\in V\setminus v_b} Cross_{v,(v_a,v_c)}(L) & \text{if } \begin{array}{l}(v_a,v_b)\in E,\\ (v_a,v_c)\in E\end{array} \end{cases}. \qquad (24)$$

(c) $Q^{dc}_{v_a,v_b}$ is a partial distance cost function of v_a and v_b, and is defined as follows:

$$Q^{dc}_{v_a,v_b}(L) = Distance_{v_a,v_b}(L). \qquad (25)$$

Similarly, $\hat{Q}^{dc}_{v_a,v_b,v_c}(\cdot)$ is a partial distance cost function of v_a, v_b, and v_c, and is defined as follows:

$$\hat{Q}^{dc}_{v_a,v_b,v_c}(L) = Distance_{v_a,v_b}(L) + Distance_{v_a,v_c}(L). \qquad (26)$$

Thus far, we found out a method to efficiently calculate Δ matrix. The purpose of extending Δ matrix is to calculate the cost difference of the swap operation. When nodes v_{α_1} and v_{α_2} are swapped, we can calculate $Swap_{v_{\alpha_1},v_{\alpha_2}}$ using these Δ costs as follows:

$$Swap_{v_{\alpha_1}v_{\alpha_2}}(S_{v_{\alpha_1}\leftrightarrow v_{\alpha_2}}L) = \Delta_{v_{\alpha_1}P(v_{\alpha_2})}(L) + \Delta_{v_{\alpha_2}P(v_{\alpha_1})}(L) + R_{v_{\alpha_1},v_{\alpha_2}}(S_{v_{\alpha_1}\leftrightarrow v_{\alpha_2}}L) - R_{v_{\alpha_1},v_{\alpha_2}}(L), \qquad (27)$$

where

$$R_{v_a,v_b}(L) = W_{ee} \sum_{e_{v_a}\in E_{v_a}, e_{v_b}\in E_{v_b}} Cross_{e_{v_a},e_{v_b}}(L) + W_{ve}\left(\sum_{e_{v_a}\in E_{v_a}} Cross_{v_b,e_{v_a}}(L) + \sum_{e_{v_b}\in E_{v_b}} Cross_{v_a,e_{v_b}}(L) \right). \qquad (28)$$

In SCCB-grid layout algorithm, the combo score also needs to be considered. Given a layout such that a node v_α is moved to a vacant point p, $\Delta^{cs}_{v_\alpha p}$ can be calculated as shown in Equation (3). In contrast, if two nodes v_{α_1} and v_{α_2} are swapped, the difference of combo scores, *Combo* ($S_{v_{\alpha_1}\leftrightarrow v_{\alpha_2}} L$) – *Combo* ($L$), is effectively calculated as follows:

$$Swap^{cs}_{v_{\alpha_1},v_{\alpha_2}}(L) = pSwap_{v_{\alpha_1}v_{\alpha_2}}(L) + pSwap_{v_{\alpha_2}v_{\alpha_1}}(L), \tag{29}$$

where

$$pSwap_{vu}(L) = \begin{cases} W_{cs}(Combo_v(S_{v\leftrightarrow u}L) - Combo_v(L) + Adj_v(S_{v\leftrightarrow u}L) - Adj_v(L) & \text{if } v \in V' \\ 0 & \text{if } v \notin V' \end{cases}. \tag{30}$$

A pseudo code of SCCB-grid layout algorithm is described in Figure 5.

If node v_β is moved at the previous step, the time complexity of calculating Δ matrix is $O\ ((|V| + |E|)|\ E_{v_\beta}\ ||U|)$. If two v_{β_1} and v_{β_2} are swapped at the previous step, the time complexity of calculating Δ matrix was $O\ ((|V| + |E|)\ (|\ E_{v_{\beta_1}}\ | + |\ E_{v_{\beta_2}}\ |)\ |U|) = O\ ((|V| + |E|)\ |\ E_{v_{\beta'}}\ ||U|)$, where $|\ E_{v_{\beta'}}\ | = (|\ E_{v_{\beta_1}}\ | + |\ E_{v_{\beta_2}}\ |)/2$. In addition, the time complexity of all the swap operations considered at each step is $O\ (|E|^2)$. Therefore, the time complexity of SCCB-grid layout algorithm is $O\ (|E|^2 + |U||\ E_{v_{\beta'}}\ |\ (|V| + |E|))$ at each step.

Since the time complexity of CB-grid layout algorithm is $O\ (|V|^2 + |E|^2 + |W||\ E_{v_\beta}\ |\ (|V| + |E|))$ at each step [15], the time complexity of SCCB-grid layout algorithm is $O(|V||\ E_{v_\beta}\ |\ (|V| + |E|))$ larger than that of CB-grid layout algorithm (note that v_β and $v_{\beta'}$ are not distinguished here). Here, we consider two cases, $|V| \leq |W|$ (case 1) and $|V| > |W|$ (case 2) and show these two algorithms have the same time complexity with high probability. For case 1, the above difference is negligible since $O\ (|V||\ E_{v_\beta}\ |\ (|V| + |E|)) \leq O\ (|W||\ E_{v_\beta}\ |(|V| + |E|))$. In contrast, the $O(|V||\ E_{v_\beta}\ |\ (|V| + |E|))$ difference cannot be neglected in case 2. However, if we assume that all nodes can be moved to form the next layout with equal probability, $|V||\ E_{v_\beta}\ | = 2\ |E|$, and $O(|V||\ E_{v_\beta}\ |\ (|V| + |E|)) = O\ (|V|^2 + |E|^2)$ subsequently. Therefore, the time complexity of SCCB-grid layout algorithm will be the same as that of CB-grid layout algorithm even in the case 2. For the above reasons, the time complexities of SCCB-grid and CB-grid layout algorithms are the same in practice.

Results and Discussion

Data and Parameters

To evaluate our algorithms on a large-scale signal transduction pathway with a gene regulatory network, we create the pathway model of an endothelial cell with Cell Illustrator [1,2] by extracting information from [19]. The model consists of 309 nodes and 371 edges (three times as large as the apoptosis model in [15], which consists of 117 nodes and 126 edges), and the maximum degree of a node is ten (eight in the apoptosis model). Grid widths and heights are fixed to 100 pixels; the total numbers of vertical and horizontal grid points are 36 and 40, respectively. We used the following information pertaining to seven GO subcellular localizations: extracellular space (GO:0005615), cytoplasm (GO:0005737), nucleus (GO:0005634), mitochondrion (GO:0005739), plasma membrane (GO:0005886), nuclear membrane (GO:0005635), and mitochondria membrane (GO:0005740). We also used the following information pertaining to sixteen processes and entities used as attributes of nodes: migration, phosphorylation, protein with a modification, ligand, assembly, transcription, translation, mRNA, ligand and receptor, receptor, unknown, protein, exchange, trimer, ubiquitination, and degradation.

Usually, these types of biological models have many nodes termed as degradation. The degradation process always has only one edge. To exploit this property, we apply these layout algorithms after removing degradation nodes (97 nodes). After applying layout algorithms, we attach each eliminated degradation node just below the entity to which it was initially connected. Thus, in practice, the numbers of nodes and edges in the model given to layout algorithms are 212 and 274, respectively. Note that when the performances of algorithms are compared with the numbers of edge-edge crossings and node-edge crossings in the latter part of this section, crossings that are caused by degradations and edges connected to them are not taken into account.

We apply the following rule to edge-edge crossing weight W_{ee}, node-edge crossing weight W_{ne}, combo score weight W_{cs}, and distance cost weight W_{dc} of a layout cost, in Equation (2), to ensure that the importance of the distance cost is less than those of the others:

$$\min(W_{ee}, W_{ne}, W_{cs}) > W_{dc} \cdot \max_{L} \sum_{v,u \in V} Distance_{v,u}(L). \tag{31}$$

In our study, W_{dc}, W_{ee}, W_{ne}, and W_{cs} were set to 1, 70, 150, and 110, respectively. Also, the constant C in CW_a was set to 12.

Using the combo score, many nodes can be aligned vertically. However, in many cases, the nodes cannot be

```
SCCB-grid layout algorithm($L$)
 1: $C_0 \leftarrow C(L)$,$\Delta_{min} \leftarrow 0$:
        $\Delta_{min}$ holds the minimum in $\Delta(L)$.
 2: makeCrossState($L$)
 3: **for each** node $v_\alpha$ and gird point $p \in U$
 4:   **if** $p$ is occupied with node $v_\gamma$ **then**
 5:     $\Delta_{v_\alpha p} \leftarrow F_{v_\alpha}(T_{v_\alpha \rightarrow P(v_\gamma)} D_{v_\gamma} L) - F_{v_\alpha}(D_{v_\gamma} L)$
 6:   **else**
 7:     $\Delta_{v_\alpha p} \leftarrow F_{v_\alpha}(T_{v_\alpha \rightarrow p} L) - F_{v_\alpha}(L)$
 8:     **if** $\Delta_{v_\alpha p} - \Delta^{cs}_{v_\alpha p} < \Delta_{min}$ **then**
 9:       $\Delta_{min} \leftarrow \Delta_{v_\alpha p} - \Delta v_\alpha p^{cs}, v_\beta \leftarrow v_\alpha,\ q \leftarrow p$
10:     **end if**
11:   **end if**
12: **end for**
13: $swap \leftarrow$ **false**
14: **for each** pair of nodes $v_{\alpha_1}$, $v_{\alpha_2}$
15:   **if** $Swap_{v_{\alpha_1}, v_{\alpha_2}} - Swap^{cs}_{v_{\alpha_1}, v_{\alpha_2}} < \Delta_{min}$ **then**
16:     $\Delta_{min} \leftarrow Swap_{v_{\alpha_1}, v_{\alpha_2}} - Swap^{cs}_{v_{\alpha_1}, v_{\alpha_2}}$,
        $v_{\beta_1} \leftarrow v_{\alpha_1}$, $v_{\beta_2} \leftarrow v_{\alpha_2}$, $swap \leftarrow$ **true**
17:   **end if**
18: **end for**
19: **do while** $\Delta_{min} < 0$
20:   $\Delta'_{min} \leftarrow 0$:
        $\Delta'_{min}$ holds the minimum
                  in $\Delta(T_{v_\beta \rightarrow q} L)$ at each step.
21:   **for each** node $v_\alpha$ and grid point $p \in U$
22:     **unless** $swap$ **then**
23:       $\Delta'_{v_\alpha p} \leftarrow \Delta'_{v_\alpha p}(T_{v_\beta \rightarrow q} L)$ using Equation (8)
24:     **else** $\Delta'_{v_\alpha p} \leftarrow \Delta'_{v_\alpha p}(S_{v_{\beta_1} \leftrightarrow v_{\beta_2}} L)$ using Equation (14)
25:     **if** $p$ is a vacant point **then**
26:       **if** $\Delta'_{v_\alpha p} - \Delta^{cs}_{v_\alpha p} < \Delta'_{min}$ **then**
27:         $\Delta'_{min} \leftarrow \Delta'_{v_\alpha p} - \Delta^{cs}_{v_\alpha p}, v'_\beta \leftarrow v_\alpha, q' \leftarrow p$
28:       **end if**
29:     **end if**
30:   **end for**
31:   $swap \leftarrow$ **false**
32:   **for each** pair of nodes $v_{\alpha_1}$, $v_{\alpha_2}$
33:     **if** $Swap_{v_{\alpha_1}, v_{\alpha_2}} - Swap^{cs}_{v_{\alpha_1}, v_{\alpha_2}} < \Delta'_{min}$ **then**
34:       $\Delta'_{min} \leftarrow Swap_{v_{\alpha_1}, v_{\alpha_2}} - Swap^{cs}_{v_{\alpha_1}, v_{\alpha_2}}$,
          $v_{\beta_1} \leftarrow v_{\alpha_1}$, $v_{\beta_2} \leftarrow v_{\alpha_2}$, $swap \leftarrow$ **true**
35:     **end if**
36:   **end for**
37:   **for each** node $v_\alpha$ and grid point $p \in U$
38:     $\Delta_{v_\alpha p} \leftarrow \Delta'_{v_\alpha p}$
39:   **end for**
40:   **unless** $swap$ **then**
41:     renewCrossState($v_\beta, L$)
42:     $v_\beta \leftarrow v'_\beta$, $q \leftarrow q'$, $\Delta_{min} \leftarrow \Delta'_{min}$
43:   **else**
44:     renewCrossStateSwap($v_{\beta_1}, v_{\beta_2}, L$)
45:     $v_{\beta_1} \leftarrow v_{\beta'_1}$, $v_{\beta_2} \leftarrow v_{\beta'_2}$, $q \leftarrow q'$, $\Delta_{min} \leftarrow \Delta'_{min}$
46:   **end if**
47: **end do**
```

Figure 5
SCCB-grid layout algorithm. A pseudo code of SCCB-grid layout algorithm.

moved once they have combo relations. Plasma membrane, nuclear membrane, and mitochondrial membrane are thin and torus shaped, thus, vertical alignments of the nodes on these subcellular localizations will not be of interest for users (e.g., the width of plasma membrane in our model is only two grids). Therefore, in this paper, we decided to ignore combo scores in plasma membrane, nuclear membrane, and mitochondrial membrane.

Comparison of layouts
Figure 6 shows the number of edge-edge crossings, the number of node-edge crossings, combo scores, and total costs of the layouts with CB-grid, CCB-grid, and SCCB-grid layout algorithms, and the human layout. We generate ten initial layouts by applying Eades initial layout algorithm to ten random layouts. These initial layouts are commonly used for each layout algorithm (CB Eades, CCB Eades, and SCCB Eades in Figure 6). In addition, we use the ten random layouts directly as initial layouts of CB-grid layout algorithms (CB random in Figure 6, which corresponds to the previous layout algorithm) to confirm the significance of preparing proper initial layouts. Figure 8 and 9 respectively show the best layouts of CB-grid and SCCB-grid layout algorithms, which have the lowest total cost among ten resulting layouts of each algorithm. The human layout is shown in Figure 10.

In [15], the initial layout for CB-grid layout algorithm was a random layout, which had a large number of edge-edge crossings and node-edge crossings. Many iterations will,

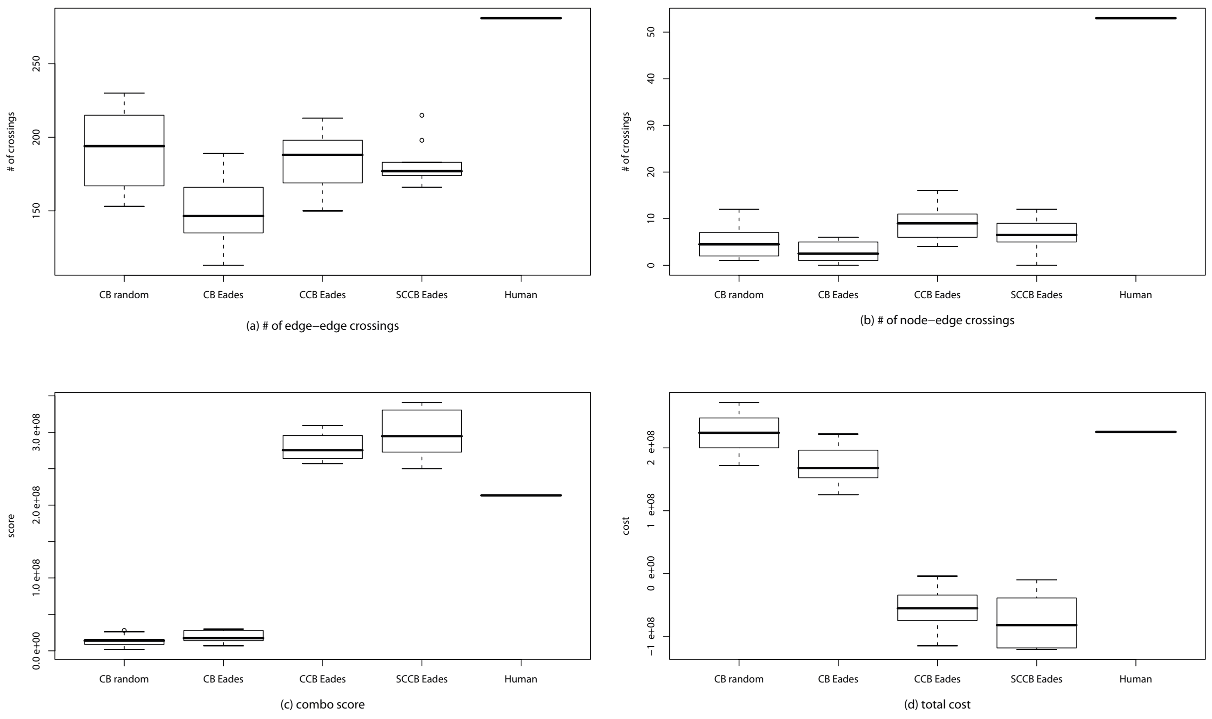

Figure 6
Comparisons of edge-edge crossings, node-edge crossings, combo score, and total cost among the results of four grid layout algorithms and the human layout. Costs and scores of the generated layouts with the CB random, CB Eades, CCB Eades, SCCB Eades, and human layout from the same initial layout. These algorithms are applied to ten initial layouts. (a) the number of edge-edge crossings. (b) the number of node-edge crossings. (c) combo score. (d) total cost.

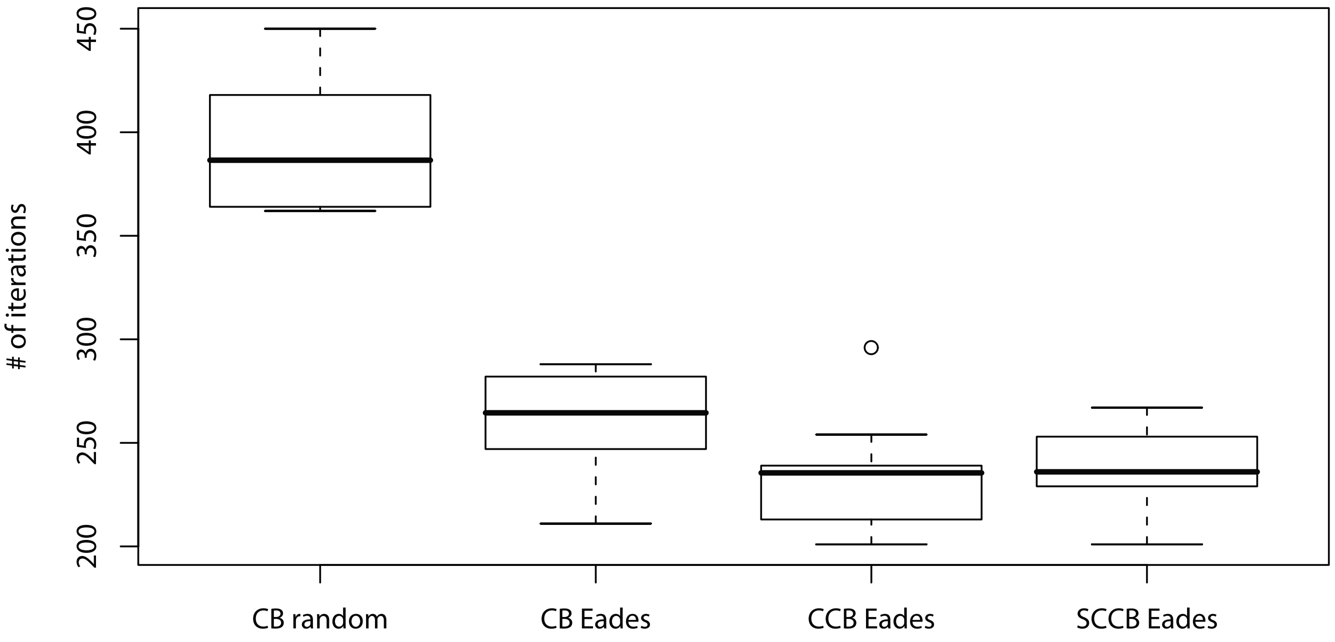

Figure 7
Comparisons of the total numbers of iterations for optimal layouts among four grid layout algorithms. Total number of iterations for optimal layouts with CB random, CB Eades, and SCCB Eades from the same initial layout. Ten initial layouts are applied with these algorithms.

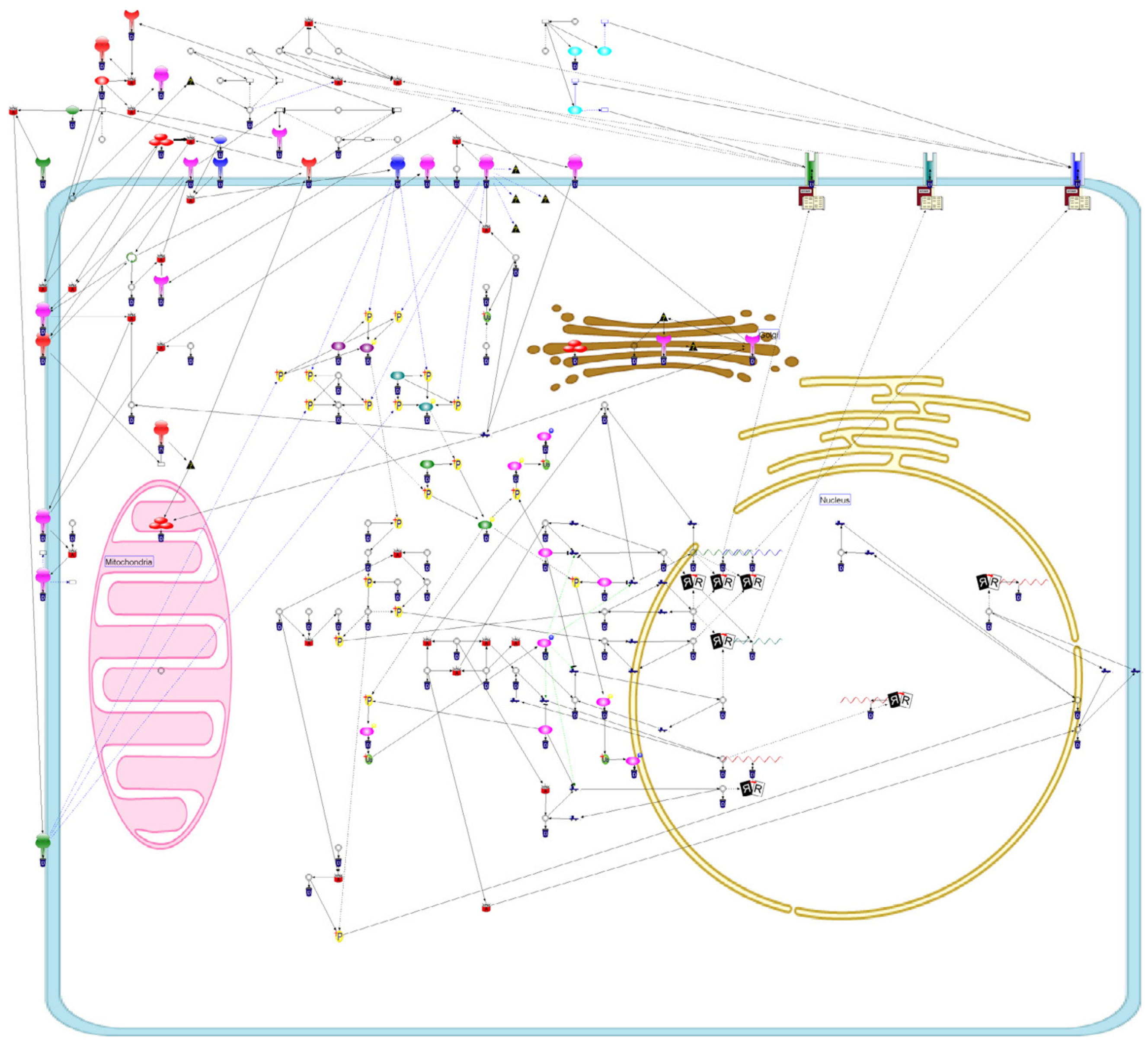

Figure 8
A resulting layout of CB-grid layout algorithm. A resulting layout of CB-grid layout algorithm in an endothelial signal transduction pathway. The pathway model is the same as that in Figure 10.

therefore, be needed until convergence. This fact prompted us to use the output of Eades initial layout algorithm as an initial layout. Figure 7 shows the number of iterations until convergence. As shown in this figure, CB-grid Eades successfully reduces the number of iterations when compared to CB-grid random (40% reduction on average). Moreover, the total score of CB-grid Eades is greatly improved over that of CB-grid random (see Figure 6(d)). A discussion in [15] was suggesting that reducing edge-edge crossings and node-edge crossings will lead to a better approximation of the human layout. In contrast as shown in Figure 6(a) and 6(b), the human layout also has several edge-edge and node-edge crossings, and has a higher combo score than that of CB-grid layout algorithm. Based on these facts, we proposed an additional scoring criterion – combo score – in CCB-grid layout algorithm. As seen through the value of combo scores (see Figure 6(c)), CCB-grid layout algorithm drastically improves this score, and this score becomes closer to that of the human layout. However, the numbers of edge-edge crossings and node-edge crossings in CCB-grid layout algorithm increase, comparing to CB-grid Eades (see Figure 6(a) and

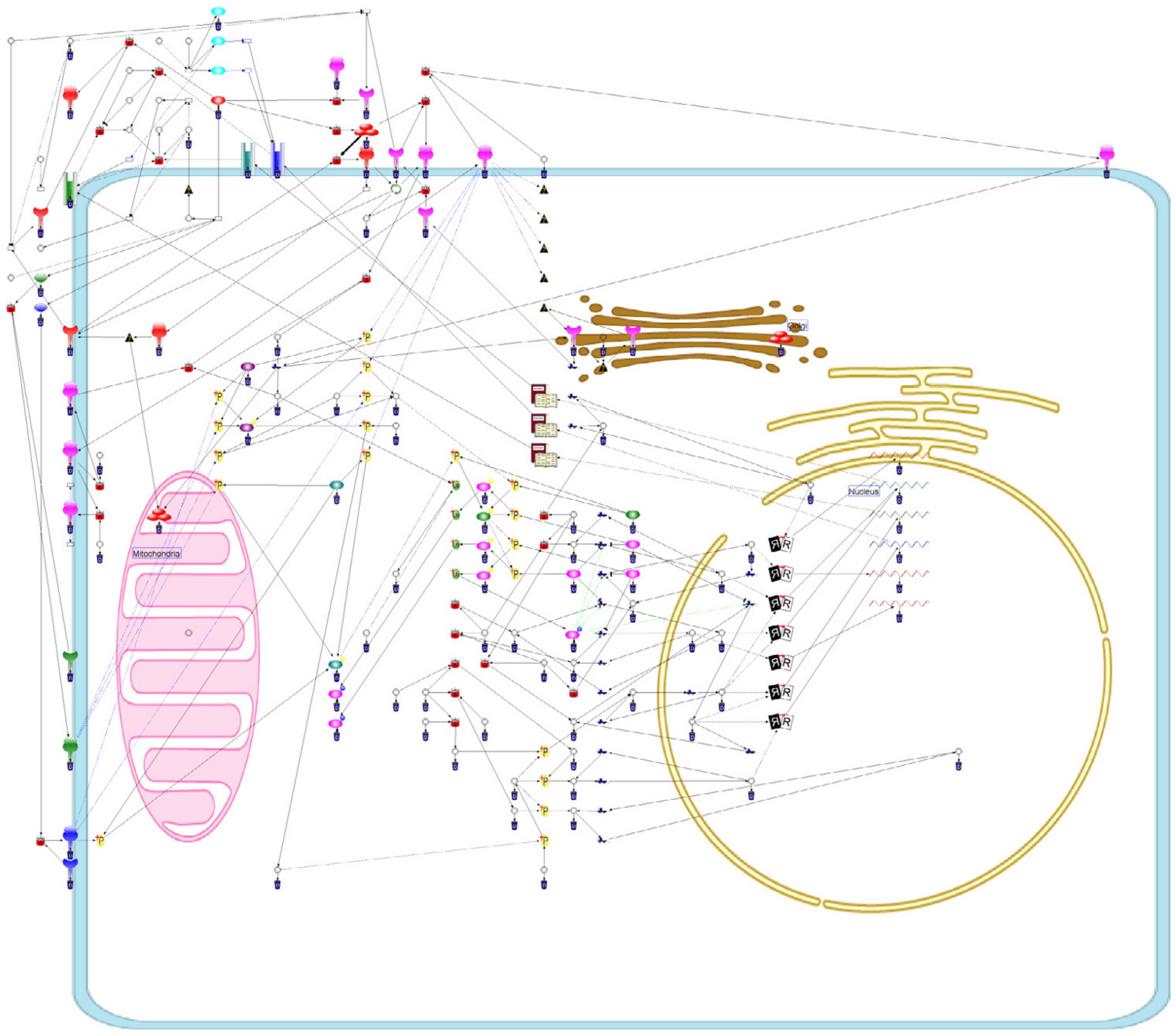

Figure 9
A resulting layout of SCCB-grid layout algorithm. A resulting layout of SCCB-grid layout algorithm in an endothelial signal transduction pathway. The pathway model is the same as that in Figure 10.

6(b)). In this paper, the swap operation is proposed to increase the number of candidate layouts at each step. As shown in Figure 6(a) and 6(b), SCCB-grid layout algorithm succeeds in reducing edge-edge crossings and node-edge crossings, i.e., the above drawback of CCB-grid layout algorithm is partially diminished. In addition, as shown in Figure 6(c), the combo score of SCCB-grid layout algorithm is also improved slightly.

We also apply grid-layout algorithms to Fas-induced apoptosis pathway model [20] and ASE cell fate simulation model [21] to obtain a more generalized comparison. Resulting layouts and the number of crossings in each layout are summarized in Additional file 1. These models including the endothelial cell model are also available as Additional file 2, and the application of SCCB-grid layout algorithm for these models can be downloaded from [22].

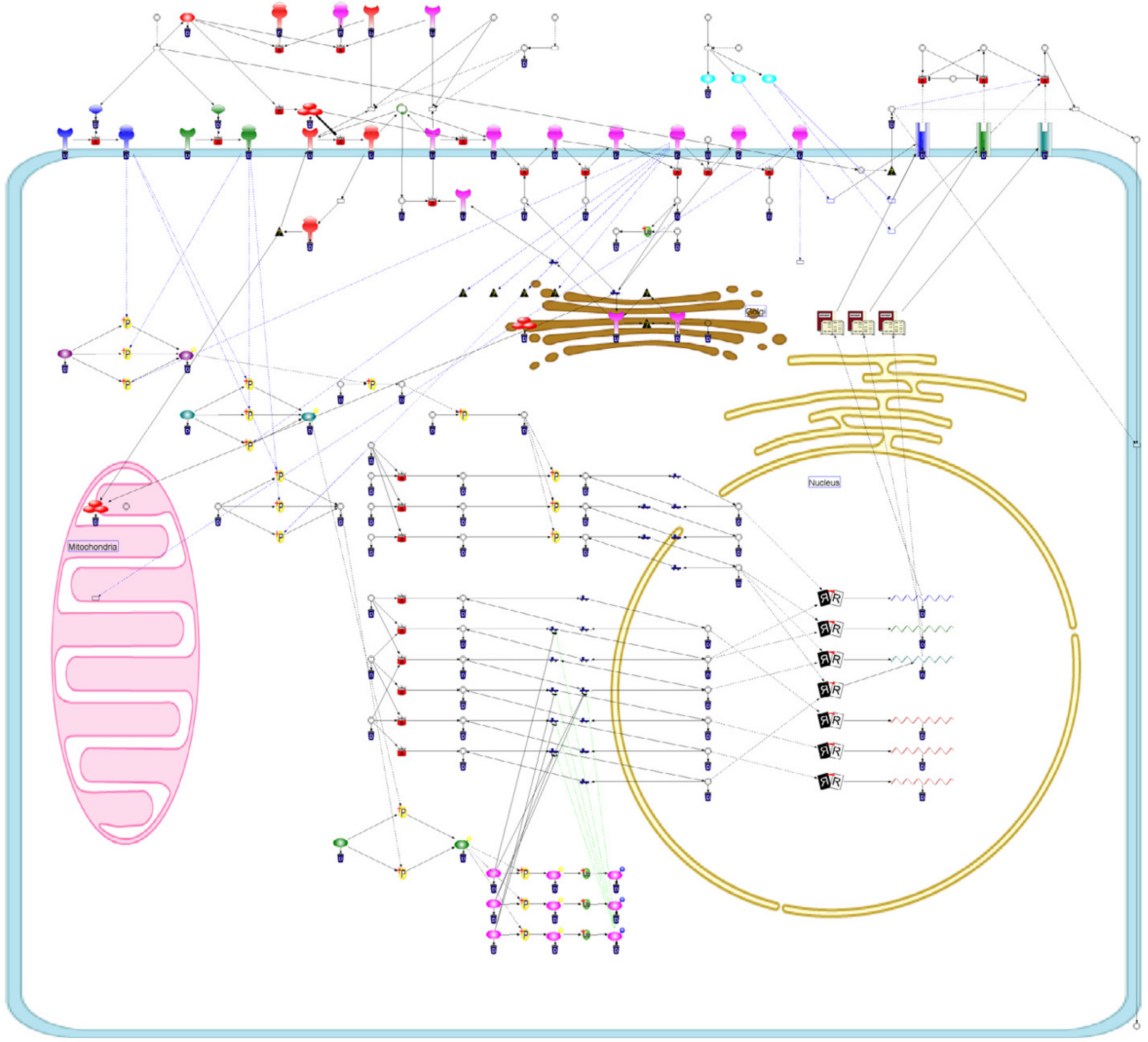

Figure 10
The human layout. The human layout of an endothelial signal transduction pathway. This pathway model is arranged with CB-grid and SCCB-grid layout algorithms in Figure 8 and Figure 9, respectively.

Conclusion

For better biopathway layouts, three improvements to CB-grid layout algorithm were proposed: (i) the improvement of initial layouts (ii) the improvement of cost function (iii) the improvement of search strategy itself without increasing the time complexity. For (i), Eades initial layout algorithm was proposed and the improvement was confirmed with a signal transduction pathway of an endothelial cell. For (ii), CCB-grid layout algorithm, which includes combo score function, was proposed and the improvement was verified with the same signal transduction pathway. For (iii), SCCB-grid layout algorithm was proposed. Due to (i) and (iii), our layout algorithm can be started from the better layout, and more robust to the condition of the initial layout than extant methods. In addition, we succeeded in utilizing the biological attributes that are not considered in extant methods due to combo score.

However, our layout algorithm has limitations and problems, which should be addressed in future work. Firstly, if the parameters of the combo score are not correctly selected, once a node gets a combo relation, the node no longer moves to other grid points anymore. Thus, it is important to devise a method that automatically selects the suitable parameters for the combo score function, edge-edge crossing function, and node-edge crossing function. Secondly, in our algorithm, only undirected graphs are considered to be laid out. On the other hand, for metabolic pathways, [11,13] proposed layout algorithms that decompose a digraph to hierarchical structural parts and directed cycle parts by considering the direction of edges in order to capture the flow of reactions. Therefore, the grid layout algorithm will also need to handle digraphs, utilizing its property that is effective especially in the grid-based layout. Finally, it should be addressed that grid layout algorithms including our new approach requires high time complexity and are not suitable for the real-time drawing. Thus, we would like to devise a further optimized grid layout algorithm to enable the real-time drawing.

Authors' contributions

The basic idea was conceived by MK and MN. This idea was developed by KK and MN who then conceived a new idea and developed it. EJ created the endothelial model in Figure 10. SM supervised the whole study. The final manuscript was read and approved by all authors.

Additional material

Additional file 1

Resulting layouts of applying LK-grid layout algorithm, CB-grid layout algorithm and SCCB-grid layout algorithm to Fas-induced apoptosis pathway model and ASE cell fate simulation model are shown. Comparison of these results are also included.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-8-76-S1.pdf]

Additional file 2

Biopathway model files. Endothelial cell model, Fas-induced apoptosis pathway model and ASE cell fate simulation model are included.
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-8-76-S2.zip]

Acknowledgements

Computation time was provided by the Super Computer System, Human Genome Center, Institute of Medical Science, University of Tokyo.

References

1. Nagasaki M, Doi A, Matsuno H, Miyano S: **Genomic Object Net: I. A platform for modelling and simulating biopathways.** *Applied Bioinformatics* 2003, **2(3):**181-184.
2. Doi A, Nagasaki M, Fujita S, Matsuno H, Miyano S: **Genomic Object Net: II. Modelling biopathways by hybrid functional Petri net with extension.** *Applied Bioinformatics* 2003, **2(3):**185-188.
3. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T: **Cytoscape: a software environment for integrated models of biomolecular interaction networks.** *Genome Research* 2003, **13(11):**2498-2504.
4. **Networks/Pajek** [http://vlado.fmf.uni-lj.si/pub/networks/pajek/]
5. Demir E, Babur O, Dogrusoz U, Gursoy A, Nisanci G, Atalay RC, Ozturk M: **PATIKA: an integrated visual environment for collaborative construction and analysis of cellular pathways.** *Bioinformatics* 2002, **18(7):**996-1003.
6. Dogrusoz U, Erson EZ, Giral E, Demir E, Babur O, Cetintas A, Colak R: **PATIKAweb: a Web interface for analyzing biological pathways through advanced querying and visualization.** *Bioinformatics* 2006, **22(3):**374-375.
7. Kurata H, Matoba N, Shimizu N: **CADLIVE for constructing a large-scale biochemical network based on a simulation-directed notation and its application to yeast cell cycle.** *Nucleic Acids Research* 2003, **31(14):**4071-4084.
8. Kurata H, Masaki K, Sumida Y, Iwasaki R: **CADLIVE dynamic simulator: direct link of biochemical networks to dynamic models.** *Genome Research* 2005, **15(4):**590-600.
9. Brandes U, Dwyer T, Schreiber F: **Visualizing related metabolic pathways in two and a half dimensions.** *Proceedings of the 11th International Symposium on Graph Drawing* 2003:111-122.
10. Karp PD, Paley SM: **Automated drawing of metabolic pathways.** *Proceedings of the 3rd International Conference on Bioinformatics and Genome Research* 1994:225-238.
11. Becker MY, Rojas I: **A graph layout algorithm for drawing metabolic pathways.** *Bioinformatics* 2001, **17(5):**461-467.
12. Sirava M, Schafer T, Eiglsperger M, Kaufmann M, Kohlgacher O, Bornberg-Bauer E, Lenhof HP: **BioMiner-modeling, analyzing, and visualizing biochemical pathways and networks.** *Bioinformatics* 2002, **18(Suppl 2):**S219-230.
13. Wegner K, Kummer U: **A new dynamical layout algorithm for complex biochemical reaction networks.** *BMC Bioinformatics* 2005, **6(212):**.
14. Li W, Kurata H: **A grid layout algorithm for automatic drawing of biochemical networks.** *Bioinformatics* 2005, **21(9):**2036-2042.
15. Kato M, Nagasaki M, Doi A, Miyano S: **Automatic drawing of biological networks using cross cost and subcomponent data.** *Genome Informatics* 2005, **16(2):**22-31.
16. Genc B, Dogrusoz U: **A constrained, force-directed layout algorithm for biological pathways.** *Proceedings of the 11th International Symposium on Graph Drawing* 2003:314-319.
17. Dogrusoz U, Gral E, Cetintas A, Civril A, Demir E: **A compound graph layout algorithm for biological pathways.** *Proceedings of the 12th International Symposium on Graph Drawing* 2004:442-447.
18. Eades P: **A heuristic for graph drawing.** *Congressus Nemerantium* 1984, **42:**149-160.
19. Pober JS: **Endothelial activation: Intracellular signaling pathways.** *Arthritis Research* 2002, **4(Suppl 3):**S109-116.
20. Matsuno H, Tanaka Y, Aoshima H, Doi A, Matsui M, Miyano S: **Biopathways representation and simulation on hybrid functional Petri net.** *In Silico Biology* 2003, **3(3):**389-404.
21. Saito A, Nagasaki M, Doi A, Ueno K, Miyano S: **Cell fate simulation model of gustatory nuerons with microRNAs double-negative feedback loop by hybrid functional Petri net with extension.** *Genome Informatics* 2006, **17:**100-111.
22. [http://www.csml.org/download/SCCBLayout_BMC_inst.exe].