國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

具骨架的超圖繪製

Hypergraph Drawing with Backbones

蔡萱尹

Hsuan-Yin Tsai

指導教授：顏嗣鈞 博士

Advisor: Hsu-Chun Yen, Ph.D.

中華民國 104 年 7 月

July 2015

國立臺灣大學碩士學位論文
口試委員會審定書

具骨架的超圖繪製
Hypergraph Drawing with Backbones

本論文係蔡萱尹君（學號 R02921040）在國立臺灣大學電機工程學系完成之碩士學位論文，於民國 104 年 7 月 24 日承下列考試委員審查通過及口試及格，特此證明

口試委員：　　（簽名）

（指導教授）

系主任、所長　　（簽名）

（是否須簽章依各院系所規定）

致謝

謝謝指導老師顏嗣鈞教授的教導，從題目的發想到整篇論文的完成都給予相當多的幫助和提點！真的很謝謝顏老師！謝謝口試委員黃秋煌教授、莊仁輝教授以及雷欽隆教授給予的建議讓這篇論文更加完整。也要特別謝謝張以潤同學的演算法教學，你真的好優秀！另外也要感謝柯有容同學、鍾家涵同學的陪伴與討論，給予我很大的信心！最後感謝我的父母支持我就讀研究所。

中文摘要

此篇論文主要在探討超圖繪製在每條超邊都有連接著一條骨架的情況下，三種美觀標準的最佳化。而三種美觀標準分別為交叉數的最小化、超邊總長度的最小化、以及在超邊互不重疊的情況下，最大數目的超邊放置量。在交叉數的最小化方面，如果有一定的順序，則可以用動態規劃演算法求解。在超邊總長度最小化方面，可以將問題轉換成 bipartite matching 去求解。另外，在超邊放置最大量方面，則用貪婪法求解。此篇論文在骨架類型方面，包含有僅有水平骨架的超圖類型、同時有水平與垂直骨架的超圖類型，以及同時有水平、垂直、斜 45 度角水平骨架的超圖類型。

關鍵字:圖形繪製，超圖，交叉數最小化、總長度最小化、骨架。

Abstract

The thesis mainly discusses various optimization problems with respect to three aesthetic criteria in hypergraph drawings under the condition that each hyperedge has one backbone. The aesthetic criteria include minimizing the number of crossings, the total hyperedge length, and maximizing the number of hyperedges under the condition that a hypergraph has no overlapping hyperedge. In the aspect of crossings minimization, if there is an order among the hyperedges, we can use a dynamic programming algorithm to solve it. In the aspect of total length minimization, we transform the problem into the bipartite matching problem to solve it. In the aspect of maximizing the number of hyperedges, a greedy algorithm is proposed. Also, the thesis considers three different types of backbones: with horizontal backbones only, with both horizontal and vertical backbones, and allowing horizontal, vertical and octilinear horizontal backbones.

Keywords: graph drawing, hypergraph, crossing minimization, length minimization, backbone.

Contents 目　錄

List of Figures

List of Tables

Chapter 1 Introduction

Information visualization is an important research topic because more and more data need to be visualized to show the pattern and make viewers understand the information in short time. To achieve this goal, many graph drawing algorithms have been proposed. For a variety of types of graphs: ordinary general graphs [1,6,8,16], hypergraphs [4,7,10,11,12,15], layered graphs [14], linear graphs [13], circular graphs [5,9], etc.

This thesis talks about drawings of hypergraphs which allow more than two nodes connected with one hyperedge. There is an additional constraint in our hypergraph drawing --- each hyperedge is associated with one backbone. Under the condition, some optimized questions which cannot be solved for ordinary graphs effectively can be solved, such as length minimization and crossings minimization. Moreover, the hyperedge number maximization problem is discussed here though this problem has not been mentioned before.

Hypergraphs has a lot of applications in practice. For example, in the electronic design automation, a circuit connects several operators, inputs or outputs [7]; in the software system visualization, software artifacts like packages, classes, methods, or attributes are represented by vertices. Edges may reflect the usage of components, call traces, method calls, or any communication between software artifacts. If one wants to present what function is committed as an edge, an edge needs to connect more than two nodes because the number of functions committed at one time is probably more than two [10]. In addition, in social networks, a hypergraph can reveal groupings of humans and their relationships for instance, modeling students of a university. Also, in the world trade graph, one might ask: Which countries are economically strongly connected to a certain country [10]?

So far some algorithms have been proposed to draw hypergraphs. Francois Perrault and Peter Eades proposed drawing hypergraphs in the subset standard [4, 11]. Instead of drawing hypergraphs in the edge standard they proposed an algorithm which considers

vertices connected with a hyperedge as a set and drawn the region to enclose these vertices like Figure 1.

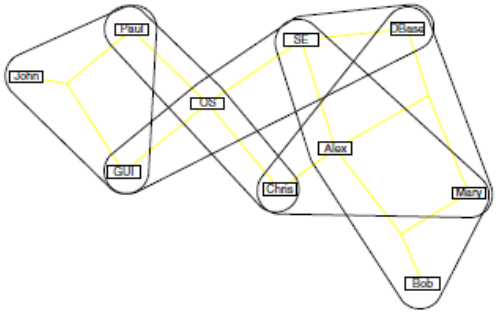


Figure 1: A hypergraph in the subset standard

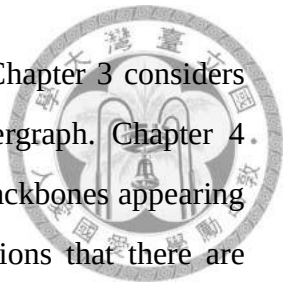
In the aspect of the edge standard, Martin Junghans proposed a method that draws hyperedges in a fixed graph layout [10]. The curves in that work are not straight-line, because node occlusions cannot be avoided. They mainly focused on how to avoid node occlusions and cluster intersections by adjusting the routing of hyperedges. And, they proposed some techniques to improve the readability of hypergraph visualization including using the concept of edge bundling.

Among aesthetic criteria in hypergraph drawing, the hypergraph crossings minimization problem (HCM) was proved as an NP-complete problem by Thomas Eschbach et al [7]. Also, they proposed a heuristic method to minimize the number of crossings by a greedy assignment, sifting (local refinement), and reordering.

Our idea of drawing a backbone within each hyperedge is motivated by the work of many-to-one boundary labeling with backbones [2]. Because one hyperedge may connect many nodes, the presence of a backbone can reduce the use of the total ink. In other words, backbones can make the display cleaner and bring less disturbing so that a viewer can clearly see which elements have relations with a certain element.

The goal of this thesis is to present several backbone styles, and to optimize some aesthetic criteria algorithmically. Our results are summarized in Table 1. The detailed description of the problem is given in Chapter 2.

The remainder of this thesis is structured as follows. The main contributions of this thesis are introduced in the following three Chapters 3, 4 and 5, which present



algorithms to fulfill the requirements of hypergraph visualization. Chapter 3 considers the condition that there are only horizontal backbones in a hypergraph. Chapter 4 considers the case that there are horizontal backbones and vertical backbones appearing in a hypergraph at the same time. Chapter 5 considers the conditions that there are octilinear horizontal backbones [3], vertical backbones, and ordinary horizontal backbones. Finally, Chapter 6 concludes this thesis and outlines areas of future work.

Table 1: The summary of results in the thesis

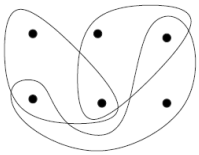
<table>
<tr><th></th><th>Horizontal</th><th>Horizontal and Vertical</th><th>Horizontal and Vertical and Octilinear</th></tr>
<tr><td>Minimization of the Number of Crossings</td><td>Fixed Order: Polynomial time</td><td>Fixed Order: Polynomial time</td><td>Fixed Order: Polynomial time</td></tr>
<tr><td rowspan="2">Minimization of the Total Length</td><td>One-degree Node: Linear time</td><td rowspan="2">Polynomial time</td><td rowspan="2">Polynomial time</td></tr>
<tr><td>Multi-degree Node: Polynomial time</td></tr>
<tr><td>Maximization of the Number of Hyperedges</td><td>Multiple Backbones in Each Gap: Polynomial time</td><td>Multiple Backbones in Each Gap: Polynomial time</td><td>Multiple Backbones in Each Gap (fixed octilinear): Polynomial time</td></tr>
</table>

Chapter 2 Preliminary

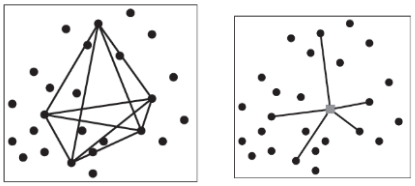
2.1 Definitions and Notations of Hypergraphs

An ordinary graph G is a pair (V, E), where V is the set of vertices and E ($\leq V \times V$) is the set of edges. A hypergraph HG = (V, H) generalizes an ordinary graph in the sense that E ($\leq 2^V$) is the set of hyperedges, each of which is a subset of vertices.

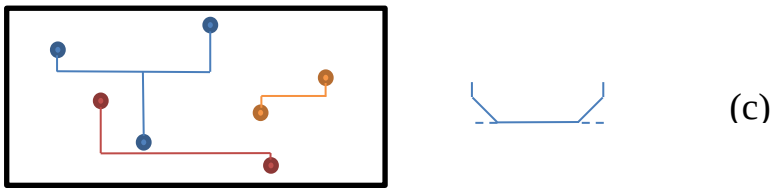
There are various types of hyperedges, such as the subset standard, the fully connected structure and the star structure of the edge standard. The hyperedge's drawing style discussed in this thesis is described in the following. Each hyperedge has only one backbone, and the vertices connected with the hyperedges by the segment which has 90 degrees angle with the backbone. The hyperedge with backbone can bring the clarity and uniformity to the graph because all vertices connected with the hyperedge must be connected with the backbone. We only have octilinear hyperedges and orthogonal hyperedges in a hypergraph drawing. Orthogonal hyperedge means that the graph can be drawn only by vertical and horizontal segments, so the hypergraph can be cleaner without any curves. And the octilinear hyperedge means that the backbone is composed of two 45 degree segments and a straight line.



(a)



(b)



(c)

Figure 2: Examples of hypergraphs: (a) Subset standard, (b) Edge standard (Left: fully connected, Right: star), (c) Hyperedges with one backbone (Left: Orthogonal, Right: Octilinear backbone)

2.2 Aesthetic Criteria

There are many different aesthetic Criteria, examples of such are:

- Minimization of the number of hyperedge crossings
- Minimization of the number of hyperedge bends
- Minimization of the total hyperedge length
- Symmetry
- Small drawing area

The aesthetic criteria discussed here are:

- Minimization of the number of hyperedge crossings
- Minimization of the total hyperedge length
- Maximization of the number of hyperedges

The object of the first criticism is to minimize the total crossings in a hypergraph drawing. A crossing may be caused by a segment and a backbone or by a backbone and the other backbone and likeness. The figure is in the following. Noted that more than one hyperedge connected with the node.

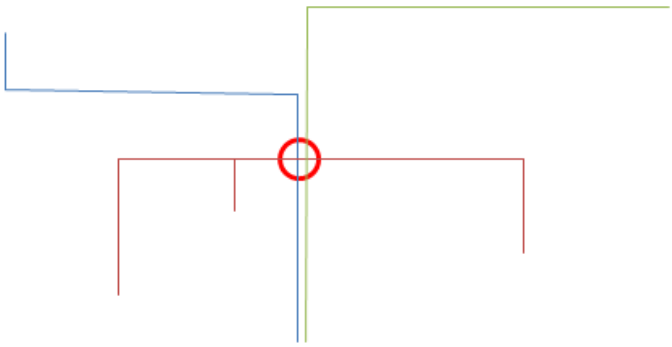


Figure 3: Example of two crossings

The object of the second criticism is to minimize the total length in a hypergraph drawing. We assume that each gap has the same height because the node placement is even, so the length can be calculated by the unit of gap's height. A hyperedge's length contains the length of a backbone and length of all segments connected with the points.

The object of the third criticism is to maximize the number of hyperedges. How do the backbones place so that a hypergraph drawing can be put as much as it can under the condition all hyperedges are overlapping-free?

2.3 Problem Description

One of the goals in information visualization area field is to guarantee that all the information is presented in a specific region such as a screen. This region is defined as a rectangular R here. All the points and edges drawn in the hypergraph must be in the rectangular R. The problem is assumed that there is a set V which includes n vertices, and the positions of them are fixed. Also, there is a set H which includes e hyperedges. The goal of this thesis is to place set V and set H in the region R, and optimize the placement of hyperedge to make hypergraph drawing good-looking.

In addition, "gap" plays an important role in this article. Backbones can only put in the gaps in the entire thesis. The gap is the space between vertices. The node position is general, so no vertex has the same x coordinate or y coordinate. For example, in the horizontal backbones situation, gaps are horizontal space between the adjacent vertically points and its length is the same as the length of the rectangular R. All backbones must be put in a gap.

Notation:

R: Restriction of drawing range.

P: Set V includes n points.

H: Set H includes e hyperedges.

Gap: In Figure 4, horizontal spaces between the vertically adjacent points. Gap 0 is above all points and Gap n is below all points.

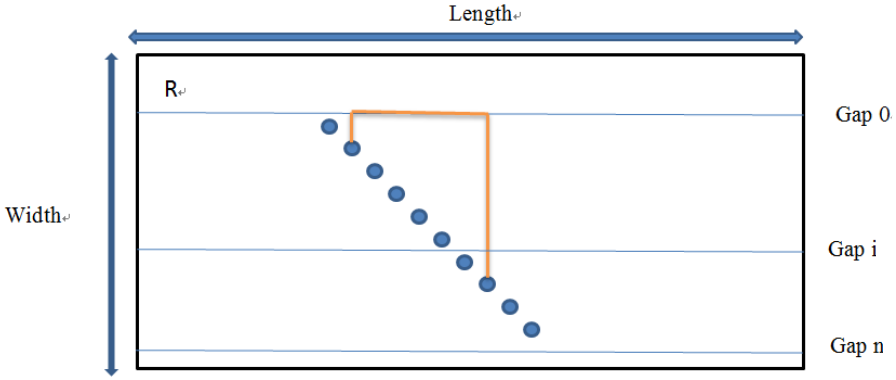


Figure 4: Illustration of problem description

Chapter 3 Algorithm Model ---- Horizontal backbones

3.1 Crossings Minimization

In this section, the y-coordinate order of hyperedges is fixed. For instance, the (i-1)-th hyperedge's backbone must be above the i-th hyperedge's backbone, and the (i+1)-th hyperedge's backbone is below the i-th hyperedge's backbone. Also, there is a constraint that only one backbone of a hyperedge is put in each gap.

Problem Definition:

Crossing minimization for hypergraphs with fixed-order horizontal backbones (HCM_fhb): Given a hypergraph HG = (V, H), an integer k, and a total order among edges in H, deciding whether there is a drawing of G such that the y coordinates of the |H| backbones respect the total order and the total number of crossings is less than or equal to k.

Theorem 1: HCM_fhb is solvable in O (|V||H|) time.

Proof: When a problem's objective is optimization, we usually think that maybe dynamic programming can solve it. The subproblems here can be got from subsubproblems, so we propose a dynamic programming to achieve the minimization. Begin at the simplest situation: the special case that the degree of each point is one. The algorithm optimizing the crossing number is below.

CROSS is a matrix to store the crossing number, and CROSS [i] [g] is the crossing number if the i-th hyperedge is placed in the g-th gap. And the main idea of iteration is that the crossing number i-th hyperedge placed in g-th gap is related to the same hyperedge placed in (g-1)th gap.

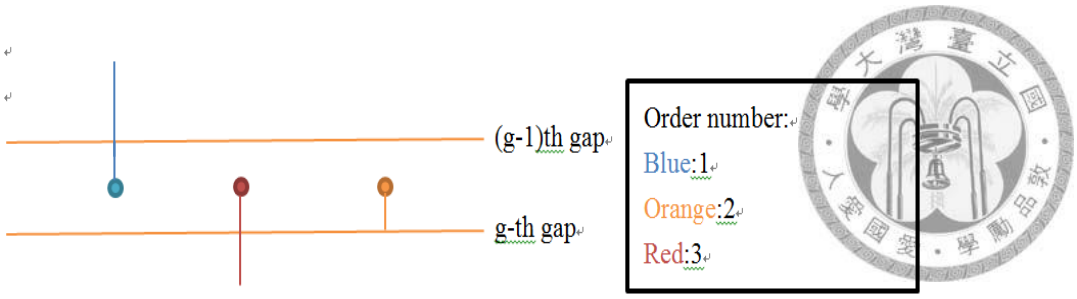


Figure 5: Illustration of HCM_fhb

Here Assumes that I want to put the i-th hyperedge (colored in orange) in the g-th gap and I defined the order number that hyperedge contains the points between gaps as j. Then

$$\text{CROSS [i] [g]} = \begin{cases} \text{CROSS [i] [g-1]+1, if j > i-th hyperedge's order} \\ \text{CROSS [i] [g-1], if j = i-th hyperedge's order} \\ \text{CROSS [i] [g-1]-1, if j < i-th hyperedge's order} \end{cases} \text{---- (1)}$$

Please note that the backbone is from the leftmost point to the rightmost point connected to hyperedge, so it is needed to check if there exists any crossing.

After filling in this CROSS table, the sum of crossing numbers needed to be calculated. SUM is a matrix to store the crossing number after already placed i hyperedges. SUM[i][g] is the sum of crossing number if it is already placed (i-1) hyperedges and the i-th hyperedge is placed in the g-th gap. Then

$$\text{SUM[i][g]} = \text{MIN}_{g'<g}\ \text{SUM[i-1,g']+CROSS[i][g]} \quad \text{----- (2)}$$

Finally, the answer is in the last i-column, and we can record the path so that knowing which hyperedge putting in which gap is the best placement. In addition, the proof of correctness is below.

Statement (1) is correct because if the order of hyperedge connected the point between two gaps is higher than the hyperedge under the recurrence, the crossing caused by the point must equal to the crossing caused by the same hyperedge in the prior gap subtracting one.

Figure 6: Illustration of different order of hypergraphs (higher order)

If the order of hyperedge connected the point between two gaps is equal to the hyperedge under the recurrence, the crossing caused by the point must equal to the crossing caused by the same hyperedge in the prior gap.

Figure 7: Illustration of different order of hypergraphs (same order)

If the order of hyperedge connected the point between two gaps is lower than the hyperedge under the recurrence, the crossing caused by the point must equal to the crossing caused by the same hyperedge in the prior gap adding one.

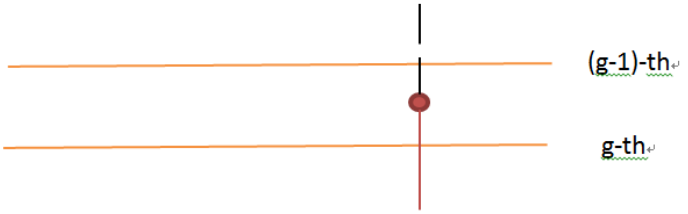


Figure 8: Illustration of different order of hypergraphs (lower order)

So, the statement 1 is correct since the recurrence considers all situations which will cause the crossing.

Statement (2) is correct and can find optimal edge cross*ings* is because of the constraint that one gap can only accommodate one backbone, the problem putting the i-th hyperedge in the g-th gap can attain from the subproblems and pre-calculated table CROSS. The minimal crossing number caused by the i-th hyperedge in g-th gap

presently would equal to the minimal value from prior results adding own crossing number.

The algorithm calculates SUM[i][g] as i increases from 1 to e and g increases from 0 to n. So, SUM[i][g] must can be calculated because the subproblems $MIN_{g'<g}$ SUM[i-1,g'] and matrix CROSS[i][g] are already complete. After accomplish this SUM table, the algorithm will return MIN_g SUM [e, g] as final answer which corresponds to the minimal number of edge crossings. □

However, the algorithm proposed here is more general. The point can be considered as a little square so that the degree of a point can more than one. The distance between the vertical segments of a hyperedge is considered extremely small.

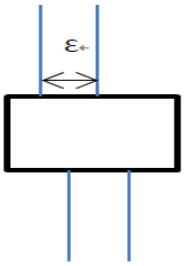


Figure 9: Illustration of a multi-degree node

As a result, the values in CROSS matrix are different. CROSS[i][g] still can be derived from CROSS[i][g-1] by more complicated calculation. The algorithm needs to find out all hyperedge connected with the point between the (g-1)-th gap and g-th gap, and check out their order. After knowing the order, the remainder part is similar as the above algorithm which solves the problem whose degree equals to one. An example is shown below.

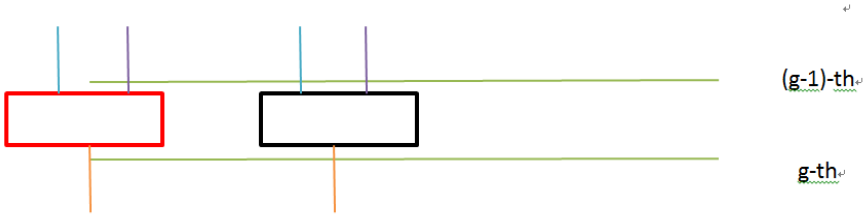


Figure 10: Illustration of a hypergraph with multi-degree nodes

Assume the order is: light blue > purple > green > orange, and then CROSS[i][g] = CROSS[i][g-1]-1. Note that the algorithm only considers that the node is entirely put in the range of backbone. The red box in the above Figure 10 will not be count in the crossing value because the crossing can be avoided.

The value of CROSS[i][g] will equal to the value of CROSS[i][g-1] adding the number of lower priority hyperedges compared to the i-th hyperedge and then subtracting the number of higher priority hyperedges compared to the i-th hyperedge.

3.2 Length Minimization

In this section, the aesthetic criteria discussed here is total length minimization, and there is a constraint that only one backbone is put in each gap. Begin at the simplest situation: the special case that the degree of each point is one. The algorithm optimizing the length is below.

Problem Description:

Length minimization for hypergraphs with horizontal backbones one degree points (HLM_hb_odp):

Given a hypergraph HG = (V, H), an integer k, deciding whether there is a drawing of HG such that the total length is less than or equal to k and its points' degrees is all one.

Theorem 1: HLM_hb_odp is solvable in O (|H|) time.

This problem can be solved by a greedy algorithm. The outside iteration begins from 1 to e. In iteration the algorithm considers that the number of node hyperedge connected is odd or even. If the number is even, the optimized location is the gaps between the two nodes close to the center hyperedge connected. In the other case the number is odd; the optimized location is the gaps above or below the middle node hyperedge connected. The algorithm is illustrated below:

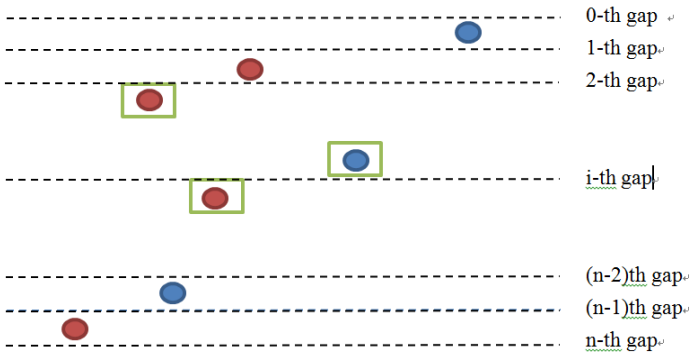


Figure 13: Example of length minimization

In Figure 13, the green boxes will decide the backbones' position. After three iterations, the final placement is shown in Figure 14.

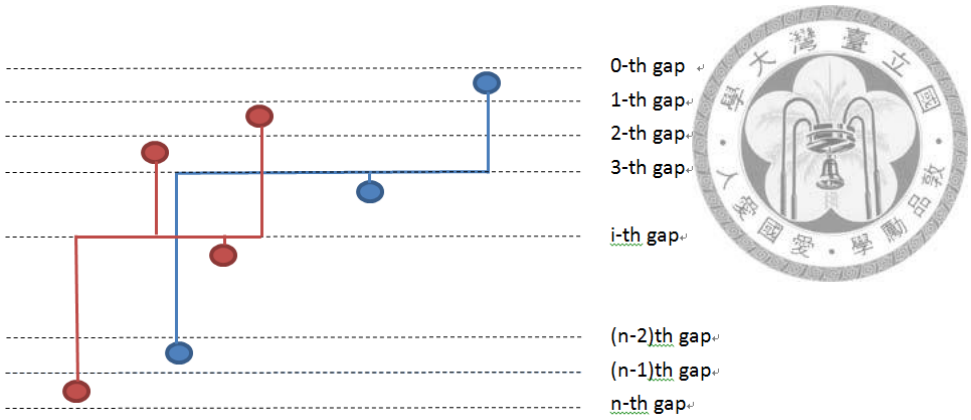


Figure 14: Example of length minimization

Every hyperedge's length is optimized. The reason is that it is observed that the minimal length will occur when backbone is put close to the center. The backbone itself' length is fixed, so the algorithm needs to choose the best position to make the vertical segments be the shortest.

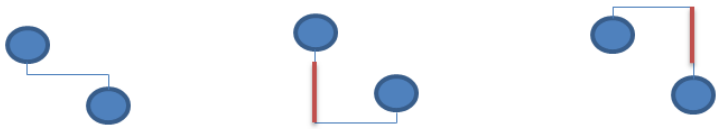


Figure 15: Illustration of cases with different lengths

In Figure 15, these red segments show how long the total length under the situation is more than the minimal length.

There are two cases: one is that the number of node hyperedge connected is odd, and the other is even.

Case 1: The optimized location is the gaps above or below the middle node hyperedge connected. For external two nodes, the best placement is between these two nodes; however, for the middle node, the best placement is under or over itself.

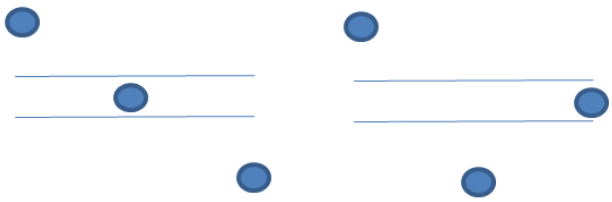


Figure 16: Example of odd nodes

Case 2: The optimized location is the gaps between the two nodes close to the center hyperedge connected.

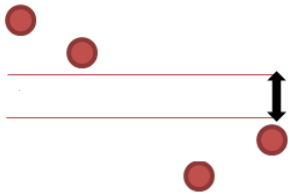


Figure 17: Example of even nodes

These optimized backbones' position can be staggered so that all backbones are in the most suitable place. This can be proved by contradiction. It is assumed that there are two optimal backbones in the same gap in the length minimization. Backbone i and backbone j are in the same gap k.

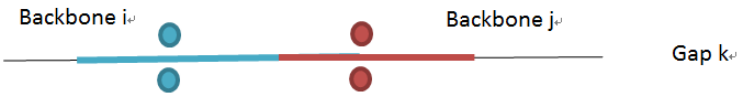


Figure 18: Example for contradiction

If backbone i must be in gap k, one of nodes is over the gap k and another node is below the gap k; and so is backbone j. Nevertheless, it is prohibited that more than one node between gaps. As a result, these optimized backbones' position will not repeat, and all backbone's positions are the best.□

However, the algorithm proposed here is more general. The point can be considered as a little square so that the degree of a point can more than one. The distance between the vertical segments of a hyperedge is considered extremely small.

Problem Description:

Length minimization for hypergraphs with horizontal backbones (LCM_hb):

Given a hypergraph HG = (V, H), an integer k, deciding whether there is a drawing of G such that the total length is less than or equal to k.

Theorem 2: LCM_hb *is solvable in O (|H|^4) time.*

As a result, the optimized backbones' position can be overlapped. In this case, it can be solved by the Hungarian Algorithm which solves the Maximum/Minimum Weight Perfect Bipartite Matching. This problem can be transformed into Minimum Weight Perfect Bipartite Matching by simple steps.

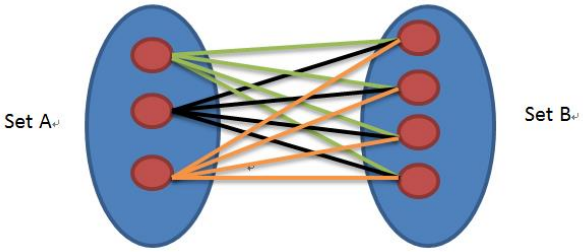


Figure 19: Bipartite matching for LCM_hb

The transformation process is below: Let every hyperedge be a point in set A, and make each gap be a point in set B. Make every point in set A be connected to every point in set B, and give the weight for the edge in the bipartite matching. The weight equals the length if the hyperedge i is put in the gap j.

The transformation is correct. It is observed that the length when a specific hyperedge is put in specific gap is fixed. The value will not change when other hyperedges are put in some gaps. Also, because the constraint here is that each gap can only accommodate one backbone, the problem can be transformed into the bipartite matching problem.□

3.3 Hyperedge Maximization

In this section, the purpose is to maximize the number of hyperedges in the hypergraph. In an ordinary situation, a hypergraph drawing can contain all hyperedges in the hyperedge set. Nevertheless, the hypergraph drawing will be filled with clutter. As a result, it is discussed under the condition that the hypergraph must be overlapping–free.

The simplest case is that only one backbone is put in each gap. The maximal number of hyperedge is n+1, because the backbone only can put in the gap and one hyperedge has one backbone. So the maximal number of hyperedges equals the number of gap, n+1.

The more general case is that each gap can accommodate multiple backbones. Given a hyperedge set of e hyperedges and (n+1) gaps with (n-1) length, optimize the number of hyperedges and make their backbone do not have any overlapping in each gap. The algorithm is in the following.

Problem Description:
Hyperedge maximization for hypergraphs with horizontal backbones (HHM_hb):
Given a hypergraph HG = (V, H), an integer k, deciding whether there is a drawing of HG such that the total number of hyperedges is larger than or equal to k.

Input: a set of e hyperedges and (n+1) gaps with (n-1) length.
Output: an overlapping-free hypergraph drawing with the largest number of hyperedges.

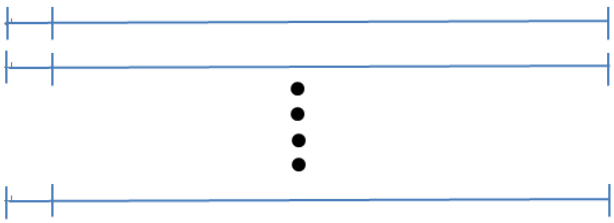
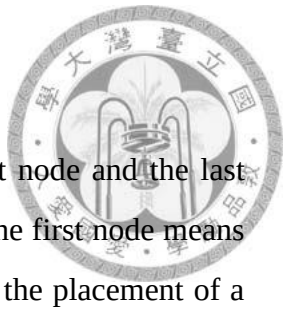


Figure 20: Illustration for HHM_hb problem description



Theorem 1: HHM_hb is solvable in O (|V|×|H|^2) time.

Firstly, classify the hyperedges in the set according to the first node and the last node which is connected with the hyperedge by vertical segments. The first node means the leftmost node, and the last node means the rightmost node; and, the placement of a backbone is decided by these two nodes. Secondly, calculate the value which means the number that other group backbones cannot placed in the gap if the backbone is placed in the same gap. Last but not least, pick the backbone whose value which calculated in the second is the minimal to place on the gap, update the value, delete the backbones cannot be put and execute the iteration with all groups of hyperedges.

And all steps above needed be execute (n+1) times, because there are (n+1) gaps that all have the same length. However, if all hyperedges are put in these gaps, the algorithm can be terminated. Every time finishing the iteration, the algorithm must delete the hyperedges that are already put well.

For example: there are six nodes in a hypergraph, which are named A, B, C, D, E, F. there are four hyperedges which are $\overline{AB}$, $\overline{ABCD}$, $\overline{ACD}$, $\overline{BDE}$.

(1) Three groups: AB, AD, BE.

(2) EXCLUDE [AB] = 1, EXCLUDE [AD] = 2, EXCLUDE [BE] = 1.

(3) Choose AB, delete AD, EXCLUDE [BE] = 1.

(4) Choose BE.

Termination: (1) ~ (4) execute (6+1) times or until all hyperedges are put.

The classified group can represent the hyperedges in iteration is because in iteration, only one hyperedges in a group can be chose. And, they are placed in the same place in the gap when their first node and the last node is the same.

The greedy algorithm can find an optimal answer. Suppose there exists the optimal solution, and the number of hyperedges of the solution is more than the greedy algorithm's result. There must exists a situation that a space that the greedy algorithm only puts one backbone, but the optimal solution puts two backbones.

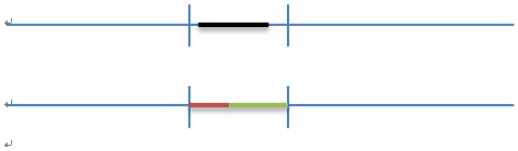


Figure 21: Illustration for greedy algorithm

Suppose the EXCLUDE [red line] = a, the EXCLUDE [green line] = b, then the EXCLUDE [black line] must be (a+b+2). The greedy algorithm should have picked the red line and the green line instead of the black line. So, the greedy algorithm is the optimal solution.□

Chapter 4

Algorithm Model --- Vertical and horizontal backbones

4.1 Crossings Minimization

4.1.1 Hypergraph with Specified backbone

In Chapter 4, the problem becomes more complex. There are vertical and horizontal backbones appearing at the same time. The backbone of a hyperedge can be vertical or horizontal, and the constraint the single gap can only accommodate one backbone is still exist. The aesthetic criterion discussed here is crossing number. The illustration of one hyperedge with a vertical backbone is in the following.

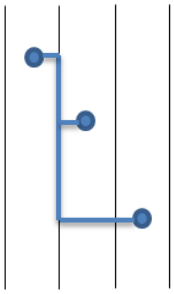


Figure 22: Illustration for vertical backbone

In this section, the order of hyperedges is fixed including the x-coordinate order and the y-coordinate order. Besides, hyperedges are split into two groups in advance.

Problem Description:

Crossing minimization for hypergraphs with fixed-order horizontal backbones, vertical backbones and hyperedges are split into two groups in advance (HCM_fhbvb_g):

Given a hypergraph HG=(V, H), an integer k, and a total order among horizontal edges and vertical edges in H, deciding whether there is a drawing of HG such that the x coordinates and y coordinates of the |H| backbones respect the orders and the total number of crossings is less than or equal to k.

Input: Given a set of n points, and its fixed position; a set of e hyperedges which includes the horizontal type and vertical type.

Constraint: classify the backbones' type in advance, single backbone in each gap.

Output: a hypergraph drawing with minimal crossings.

Theorem 1: HCM_fhbvb_g *is solvable in O (* $\frac{|V|}{2}$^2 × ($\frac{|H|}{2}$)^2*) time.*

The situation is more complicated than the situation in Chapter 3. The causes of crossings are four: two of these causes are caused by the same direction backbones, vertical and vertical, horizontal and horizontal; the other two causes are caused by the different direction backbones, and the illustration is below.

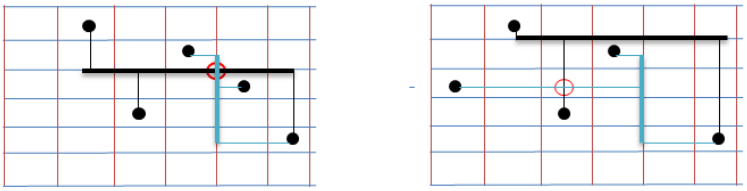


Figure 23: Example of crossings

The CROSS matrix records the crossing number that the i-th backbone is put in the g-th gap (horizontal) and the j-th backbone is put in the G-th gap (vertical). The SUM matrix records that the minimal crossing number when the i-th backbone is put in the g-th gap (horizontal) and the j-th backbone is put in the G-th gap (vertical) and the preceding (i-1) horizontal backbones and (j-1) vertical backbones are put already.

The value of CROSS matrix is similar with the one in section 3.1.1.

CROSS[i][g][j][G] = CROSS_SAMEH[i][g] + CROSS_SAMEV[j][G] (crossings caused by the same direction (like CROSS[i][g] in section 3.1.1)) + CROSS_M[i][g][j][G] (crossings caused by the two hyperedges (the i-th backbone and the j-th backbone)).

SUM[i][g][j][G] = $\text{MIN}_{g'<g,G'<G}$ (SUM[i-1][g'][j-1][G']) +

$\sum_{j'=0}^{j'=j-1}$ CROSS_M[i][g][j′][the gap from the min SUM] + ----- (1)

$\sum_{i'=0}^{i'=i-1}$ CROSS_M[i′][the gap from the min SUM][j][G] + CROSS[i][g][j][G]

Statement (1) is correct. The minimal crossing number can be divided into two parts. One part is fixed, and it contains crossings caused by the i-th hyperedge and the j-th hyperedge and crossings caused by the same direction backbones. The second part changes according to the gap's order so that the algorithm can choose minimal crossing number, and it includes the crossing caused by the preceding (i-1) horizontal backbones and the preceding (j-1) vertical backbones, the crossings caused by the i-th backbone and the preceding (j-1) vertical backbones, and the crossings caused by the j-th backbone and the preceding (i-1) horizontal backbones. The main idea is that the causes of crossings are independent.

The algorithm can find optimal answers. The SUM table will always return the optimal solution. And, all value can be calculated by the pre-computed cells. The final answer will be shown in the

$\text{MIN}_{g,G}$ SUM[the number of horizontal hyperedges][g][the number of vertical hyperedges][G].□

4.1.2 Hypergraph with Unspecified backbone

In this section, the only difference from the section 4.1.1 is that the hyperedges don't be spilt in two groups in advance. Still, the hyperedges have specific order. For instance, the (i-1)-th hyperedge's backbone is above the i-th hyperedge's backbone; the (j-1)-th hyperedge's backbone is on the j-th hyperedge's backbone's left.

Problem Description:

Crossing minimization for hypergraphs with fixed-order horizontal backbones and vertical backbones (HCM_fhbvb):

Given a hypergraph HG=(V, H), an integer k, and a total order among horizontal edges and vertical edges in H, deciding whether there is a drawing of HG such that the x coordinates and y coordinates of the |H| backbones respect the orders and the total number of crossings is less than or equal to k.

Input: Given a set of n points, and its fixed position; a set of e hyperedges which includes the horizontal type and vertical type.

Constraint: single backbone in each gap.

Output: a hypergraph drawing with minimal crossings.

Theorem 1: HCM_fhbvb is solvable in O ($\frac{|V|}{2}$ ^2 × ($\frac{|H|}{2}$)^2 × |H|) time.

The backbone being horizontal or vertical is decided by the dynamic programming.

If the backbone is horizontal:

A (i) = SUM[i][g][j][G] = $\text{MIN}_{g'<g,G'<G}$ (SUM[i-1][g'][j][G']) +

$\sum_{j\prime=0}^{j'=j}$ CROSS_M[i][g][j′][the gap from the min SUM] + CROSS_SAMEH[i][g]

If the backbone is vertical:

B (j) = SUM[i][g][j][G] = $\text{MIN}_{G'<G.g'<g}$ (SUM[i][g'][j-1][G']) +

$\sum_{i\prime=0}^{i\prime=i}$ CROSS_M[i′][the gap from the min SUM][j][G] + CROSS_SAMEV[j][G]

```
Backbones type assign () {
    for k = 0 to e do {
        Answer += MIN (A (k) or B (k))
    }
}
```

The proof is similar as the one in section 4.1.1.

4.2 Length Minimization

In this section, the aesthetic criteria discussed here is total length minimization, and there is a constraint that only one backbone is put in each gap no matter it is vertical or horizontal. The order of hyperedges is flexible and the degree of nodes can be not only one. The algorithm optimizing the length is below.

Length minimization for hypergraphs with horizontal backbones and vertical backbones (HLM_hbvb):
Given a hypergraph HG = (V, H), an integer k, deciding whether there is a drawing of HG such that the total length is less than or equal to k.

Theorem 1: HLM_hbvb is solvable in O (|H|^4) time.
Proof: There are two cases, one is that the group is split in advance and the other doesn't.
Specified case:

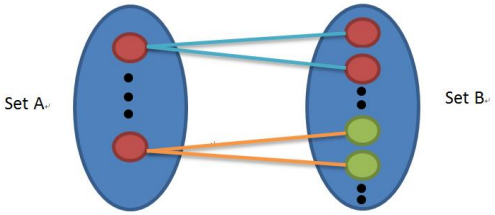


Figure 24: Bipartite matching for specified case

We transform our problem to a bipartite matching; the nodes in set A correspond to all hyperedge's backbone; the red nodes in set B correspond to the gaps in horizontal and the green nodes correspond to the gaps in vertical. Because the group type is already known, the edge between two sets is connected from horizontal type backbone to the gaps in horizontal and vice versa. Each edge has weight which presents its length.

Unspecified case:

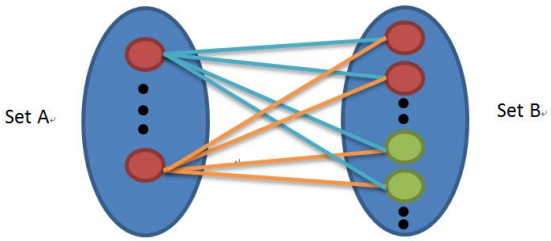


Figure 25: Bipartite matching for Unspecified case

The difference from the specific case is the edges between two sets. All edges are connected because we don't know whether the backbone being vertical or horizontal is better. After the transformation, the problem becomes the Minimum Weight Perfect Bipartite Matching which can be solved in polynomial time.

The transformation is correct. It is observed that the length when a specific hyperedge is put in specific gap is fixed. The value will not change when other hyperedges are put in some gaps. Also, because the constraint here is that each gap can only accommodate one backbone, the problem can be transformed into the bipartite matching problem.□

4.3 Hyperedge Maximization

In this section, the purpose is to maximize the number of hyperedges on the hypergraph. In an ordinary situation, a hypergraph drawing can contain all hyperedges in the hyperedge set. Nevertheless, the hypergraph drawing will be filled with clutter. As a result, it is discussed under the condition that the hypergraph must be overlapping –free.

Input: $2 \times (n + 1)$ gaps with $n - 1$ length, a set of e hyperedges which contains half horizontal backbones and half vertical backbones.
Constraint: single backbone in each gap.
Output: an overlapping-free hypergraph drawing with maximal hyperedge number.

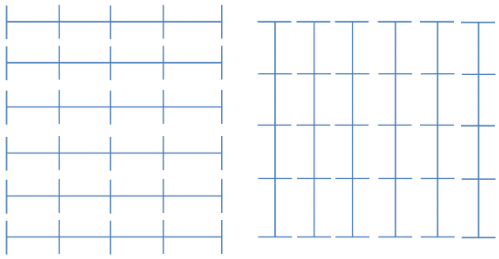


Figure 26: Illustration for HHM_hbvb

The simplest case is that only one backbone is put in each gap. The maximal number of hyperedge is $2 \times (n + 1)$, because the backbone only can put in the gap and one hyperedge has one backbone. So the maximal number of hyperedges equals to the number of gap, $2 \times (n + 1)$.

Input: $2 \times (n + 1)$ gaps with $n - 1$ length, a set of e hyperedges which contains half horizontal backbones and half vertical backbones.
Constraint: classify the type of backbones of hyperedges in advance.
Output: an overlapping-free hypergraph drawing with maximal hyperedge number.

The more general case is that each gap can accommodate multiple backbones. Given a hyperedge set of e hyperedges and $2 \times (n + 1)$ gaps with $n - 1$ length, optimize the number of hyperedges and make their backbone do not have any overlapping in each gap. The algorithm is in the following.

The hyperedges are classified in advance, so separate them in the beginning. In the aspect of the horizontal backbones, put them into the horizontal gaps. It is means that the algorithm here is the same as the one in section 3.3; in the aspect of the vertical backbones, put them into the vertical gaps. The only difference from the algorithm in section 3.3 is that the space they occupied may not be the same. In other word, the length of a vertical hyperedge is different from the length of the horizontal hyperedges even that they are connected with same nodes. And, the length is an important factor that will impact the number of the hyperedge we can put in a hypergraph drawing.

Problem Description:

Hyperedge maximization for hypergraphs with horizontal backbones and vertical backbones (HHM_hbvb):

Given a hypergraph GH = (V, H), an integer k, deciding whether there is a drawing of HG such that the total number of hyperedges is larger than or equal to k.

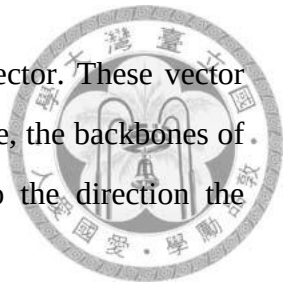
Input: $2 \times (n + 1)$ gaps with $n - 1$ length, a set of e hyperedges which contains half horizontal backbones and half vertical backbones.

Output: an overlapping-free hypergraph drawing with maximal hyperedge number.

Theorem: HHM_hbvb is solvable in O (|V|×|H|^2) time.

Proof: The outside iteration is from 0 to n. In iteration, there two gaps needed to be filled with, one is horizontal, and the other is vertical. Using two vectors to record the value that the number of backbones that cannot be put: EXCLUDE_H records that if the hyperedge is put in a horizontal gap, the number that other backbones cannot be put in the same gap; EXCLUDE_V records that if the hyperedge is put in a vertical gap, the number that other backbones cannot be put in the same gap. The algorithm will choose

the minimal value of both the EXCLUDE_H and EXCLUDE_V vector. These vector also save which backbones cannot be put. After picking the backbone, the backbones of the EXCLUDE_H or EXCLUDE_V will be deleted according to the direction the backbone was put. If the hyperedge is put already, record it.



For example: there are six nodes in a hypergraph, which are named A, B, C, D, E, F. The order in the horizontal is A, B, C, D, E, F, and the order in the vertical is E,C,A,B,D,F.

There are four hyperedges which are $\overline{AB}$, $\overline{ABCD}$, $\overline{ACD}$, $\overline{BDE}$.

(1) EXCLUDE_H [$\overline{AB}$] = 2, EXCLUDE_H [$\overline{ABCD}$] = 3, EXCLUDE_H [$\overline{ACD}$] = 3, EXCLUDE_H[$\overline{BDE}$] = 2.

(2) EXCLUDE_V [$\overline{AB}$] = 3, EXCLUDE_V [$\overline{ABCD}$] = 3, EXCLUDE_V [$\overline{ACD}$] = 3, EXCLUDE_V[$\overline{BDE}$] = 3.

(3) Choose $\overline{AB}$ to put in the horizontal gap, delete $\overline{ABCD}$, $\overline{ACD}$ of EXCLUDE_H, EXCLUDE_H [$\overline{BDE}$] = 2, EXCLUDE_V [$\overline{ABCD}$] = 2, EXCLUDE_V [$\overline{ACD}$] = 2, EXCLUDE_V[$\overline{BDE}$] = 2. $\overline{AB}$ is put already.

(4) Choose $\overline{BDE}$ to put in the horizontal gap. EXCLUDE_V [$\overline{ABCD}$] = 1, EXCLUDE_V [$\overline{ACD}$] = 1. $\overline{BDE}$ is put already.

(5) Choose $\overline{ABCD}$ to put in the vertical gap. $\overline{ABCD}$ is put already.

Termination: when execute (6+1) times or until all hyperedges are put.

The greedy algorithm can find one optimal placement. Suppose there is an optimal solution. There are two situations that that the largest number of my greedy algorithm is less than optimal solution. Assume greedy algorithm picks the k-th hyperedge's backbone to put in the horizontal gap.

Situation (1): the optimal solution picks the same backbone to the vertical gap.

Figure 27: Illustration for situation (1)

Because the optimal solution should have more hyperedges, the space originally for the k-th backbone must be filled with another backbone such as the orange line. However, the EXCLUDE_H[orange line] must be less than EXCLUDE_H[black line]. The greedy algorithm should have picked the orange line instead of black line.

Situation (2): the optimal solution doesn't pick the same backbone to the vertical gap nor horizontal gap.

Figure 28: Illustration for situation (2)

Because the optimal solution should have more hyperedges, the space originally for the k-th backbone must be filled with another backbone such as the orange line. However, the EXCLUDE_H[orange line] must be less than EXCLUDE_H[k-th hyperedge]; EXCLUDE_V[orange line] must be less than EXCLUDE_V[k-th hyperedge]. The greedy algorithm should have picked the orange line instead of the k-th hyperedge. □

Chapter 5

Algorithm Model ---- Octilinear backbones

5.1 Cross Minimization

5.1.1 Hypergraph with Fixed octilinear segment length

The meaning of the word “octilinear” is that the slope of each segment of an edge path is a multiple of 45 degree. An octilinear backbone is composed of two 45 degree segments and a straight line [3].

Figure 29: Illustration for octilinear backbone

Figure 29(a) shows that a horizontal backbone and two vertical segments; Figure 29(b) shows that two 45 degree segments connected with a horizontal backbone. This chapter discusses the hypergraph visualization that consists of horizontal backbones, vertical backbones and horizontal octilinear backbones. The reason why the hypergraph only contains the horizontal octilinear backbone is that the hyperedge may overlap each other if both octilinear backbones (vertical and horizontal) are used. And in the section 5.1, we want to minimize the crossing number with the constraints: single backbone in each gap, fixed order, and specified group.

In this section, every 45 degree segments’ length equals to $\sqrt{2} \times (d)$, and the d is the distance between the gap and the node in the gap. The illustration is below.

Problem Description:

Crossing minimization for hypergraphs with fixed-order horizontal backbones, vertical backbones and octilinear horizontal backbones; and fixed octilinear length (HCM_fhbvbo_fol):

Given a hypergraph HG=(V, H), an integer k, and a total order among edges in HG, deciding whether there is a drawing of G such that the x,y coordinates of the |H| backbones respect the total order and the total number of crossings is less than or equal to k.

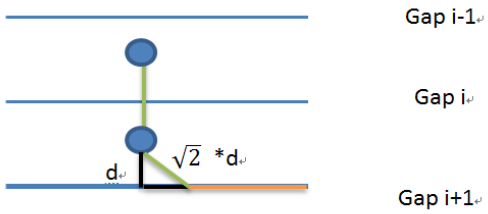


Figure 30: Illustration for HCM_fhbvbo_fol

Theorem: HCM_fhbvbo_fol is solvable in O ($\frac{|V|}{2}$^2 × ($\frac{|H|}{2}$)^2) time.

Proof: the algorithm is as same as the algorithm in section 4.1.1.

The minimal crossing number is the same as the one in section 4.1.1. Because the crosses that octilinear backbone cause is always same as the hypergraph drawing without octilinear backbones. No matter using octilinear backbones or not, the crossing number does not change; it will not increase or decrease. In the following, we will list all possible case.

Case 1: horizontal octilinear backbone CROSS horizontal backbone

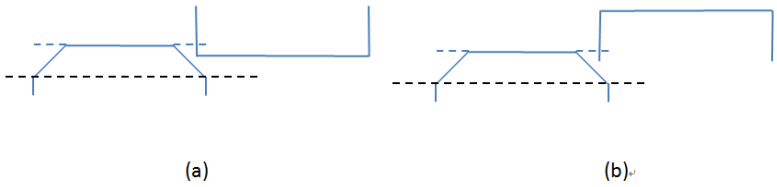


Figure 31: Example for horizontal octilinear backbone CROSS horizontal backbone

Figure 31(a): because the length of the octilinear segment.

Figure 31(b): because the node's position. (The black dotted line)

Case 2: horizontal octilinear backbone CROSS vertical backbone

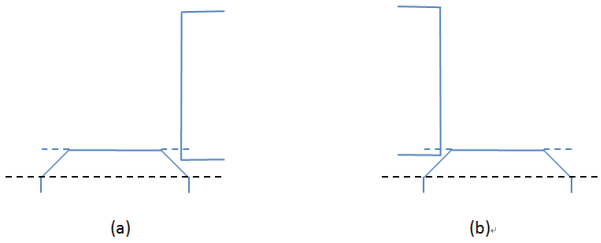


Figure 32: Example for horizontal octilinear backbone CROSS vertical backbone

Figure 32(a): because the node's position.

Figure 32(b): because the node's position.

If there are any cases that can reduce the crossing number, the minimal crossing might happen when using an octilinear backbone. If not, the minimal crossing number can also appear under the condition in Chapter 4.□

5.1.2 Hypergraph with Flexible octilinear segment length

In this section, the length of each 45 degree segment is flexible.

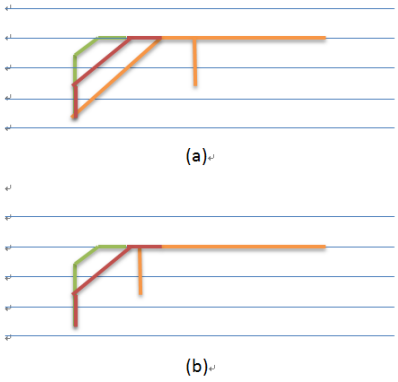
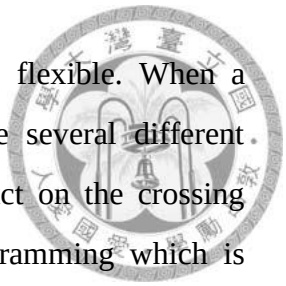


Figure 33: Example of flexible octilinear backbone

In Figure 33(a), the length can be at most $\sqrt{2} \times 2.5$ gap's length, and it can be $\sqrt{2} \times 1.5$ or $\sqrt{2} * 0.5$ gap's length too. In figure (b), the length can be $\sqrt{2} \times 1.5$ or $\sqrt{2} * 0.5$ gap's length, because there exists a vertical segment needed to be connected with in the middle of the backbone. In these two figures, the other side's drawing is identical to left side. (If the left is drawn in red form, the right side must be drawn in red form too).

Input: a set of n points and its fixed position, a set of e hyperedge and its fixed order, its specified type (vertical or horizontal or octilinear).
Constraint: single backbone in each gap, classify the backbones in advance.
Output: the hypergraph drawing with minimal crossing number.

The main idea is that the crossing number is fixed when the hyperedge's order is appointed. When the k-th hyperedge is put in the g-th gap, the minimal crossing number right now will be the preceding minimal crossing number plus its own crossing number. And, its own crossing number can be calculated while i hyperedges are placed in horizontal; j hyperedges are placed in vertical; and o horizontal hyperedges are octilinear.



However, the o horizontal hyperedges' octilinear segment is flexible. When a hyperedge is given, the horizontal octilinear backbone may have several different lengths like the figure shown above. And, every option has impact on the crossing number. Generally, these can also be solved by a dynamic programming which is similar with the one in section 4.1.2.

The length of an octilinear horizontal backbone is decided by the dynamic programming, and the largest number of length possibilities is |V|.

Problem Description:

Crossing minimization for hypergraphs with fixed-order horizontal backbones, vertical backbones, and octilinear horizontal backbones; and flexible octilinear length (HCM_fhbvbo):

Given a hypergraph HG=(V, H), an integer k, and a total order among horizontal edges and vertical edges in H, deciding whether there is a drawing of HG such that the x coordinates and y coordinates of the |H| backbones respect the orders and the total number of crossings is less than or equal to k.

Theorem 1: HCM_fhbvbo is solvable in O ($\frac{|V|}{2}$*^2×* $\frac{|H|}{2}$*^2× |H|× L) time, where L is largest number of possibilities for each octilinear hyperedge's length.*

Proof: the algorithm is similar as the one in section 4.1.2.

A[l] [k] is minimal crossing numbers when the k-th backbone's octilinear length is l.

A[k] is minimal crossing numbers as the one in section 4.1.1.

HCM_fhbvbo (){

for (k=0; k<n; k++) {

if the k-th backbone is octilinear, the possible length is l {

Answer += MIN (A[l] [k])

}

Else {

if the backbone is horizontal:

A (i) = SUM[i][g][j][G] = $MIN_{g'<g,G'<G}$ (SUM[i-1][g'][j][G']) +

$\sum_{j\prime=0}^{j'=j}$ CROSS_M[i][g][j′][the gap from the min SUM] + CROSS_SAMEH[i][g]

if the backbone is vertical:

B (j) = SUM[i][g][j][G] = $MIN_{G'<G,g'<g}$ (SUM[i][g'][j-1][G']) +

$\sum_{i\prime=0}^{i\prime=i}$ CROSS_M[i′][the gap from the min SUM][j][G] + CROSS_SAMEV[j][G]

Answer += MIN (A (k) or B (k))

}

}

}

5.2 Length Minimization

Problem Description:

Length minimization for hypergraphs with horizontal backbones, vertical backbones, and horizontal octilinear backbones (HLM_hbvbo):

Given a hypergraph HG = (V, H), an integer k, deciding whether there is a drawing of HG such that the total length is less than or equal to k.

Input: a set of n points and its fixed position, a set of e hyperedges which consist of fixed horizontal octilinear length backbones, ordinary horizontal backbones and vertical backbones with flexible order.

Constraint: single backbone in each gap, specified group.

Output: hypergraph drawing with the minimal total hyperedge length.

Theorem 1: HLM_hbvbo is solvable in O (|H|^4) time.

Proof: First, we consider the specified case which means the hyperedges are appointed to the one of the three types of backbones. Make every hyperedge be the node in set A, and make each gap be the node in set B. The problem can be transformed into a bipartite matching problem. The weight of the edge between the set A and the set B is calculated beforehand, and the value of the weight is the length while the hyperedge is put in the gap.

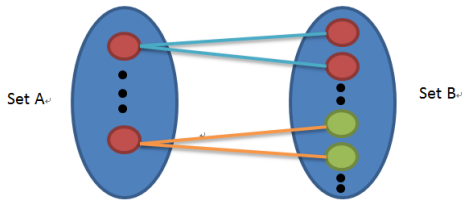


Figure 34: Bipartite matching for specified case

The red nodes in set B represent the horizontal gap, and the green nodes represent the vertical gap. In specified case, the edge is connected with according to their type. As an octilinear type backbone, the length may less than the ordinary one. It is the only difference between the section 4.2 and this section.

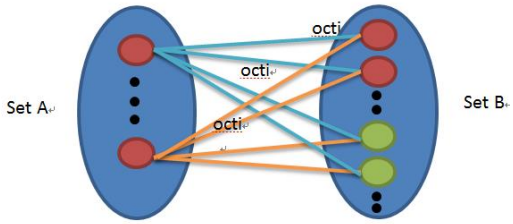


Figure 35: Bipartite matching for unspecified case (fixed octilinear length)

Secondly, if the types are not appointed in advance, the algorithm will force all horizontal backbones to become octilinear to make the total length minimal, and then solve it by the algorithm that solves Minimum Weight Perfect Bipartite Matching.

Lastly, if the length of octilinear segments is flexible, the weight of the edge of the matching will be multiple.

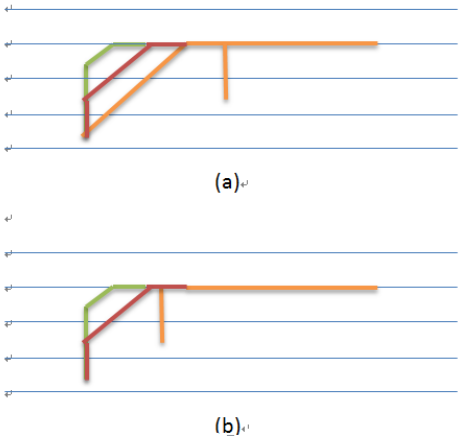
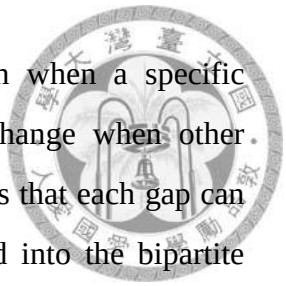


Figure 36: Example for flexible octilinear length

For instance, the length when the hyperedge is put in the appointed gap is different according to the color in Figure 36(a). However, the only thing we care about is the length; the minimal length is the ultimate goal. The algorithm will make optimized choice among these options. It will choose orange backbone in Figure 36(a), and choose red backbone in Figure 36(b). The subsequent processing is like the above.



The transformation is correct. It is observed that the length when a specific hyperedge is put in specific gap is fixed. The value will not change when other hyperedges are put in some gaps. Also, because the constraint here is that each gap can only accommodate one backbone, the problem can be transformed into the bipartite matching problem.□

5.3 Hyperedge Maximization

Input: $2 \times (n + 1)$ gaps with $n - 1$ length, a set of e hyperedges which contain the horizontal octilinear backbones, ordinary horizontal backbones and vertical backbones.

Constraint: fixed octilinear length.

Output: an overlapping-free hypergraph drawing with maximal hyperedge number.

Problem Description:

Hyperedge maximization for hypergraphs with horizontal backbones and vertical backbones and fixed horizontal octilinear backbones (HHM_hbvbo_fol):

Given a hypergraph HG = (V, H), an integer k, deciding whether there is a drawing of HG such that the total number of hyperedges is larger than or equal to k.

Theorem 1: HHM_hbvbo_fol *is solvable in O (|V|×|H|^2) time.*

Proof: We firstly show that the situation that may increase the number of hyperedges in the following illustration.

Fixed length case: in the original drawings, the blue hyperedge and the orange hyperedge will overlap each other if they are placed in the same gap. The black segment is the place that the blue hyperedge's backbone will occupy originally.

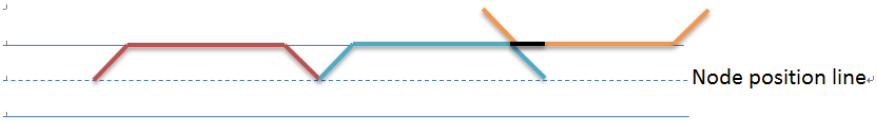


Figure 37: Example for Fixed length case

The algorithm can calculate the length of backbone when the type is already known. And the subsequent processing is like the algorithm in the second case of section 4.3. In the aspect of vertical gaps, it is all the same. In the aspect of horizontal gaps, the definition of the gradation on the gaps needs to change slightly, because the octilinear type's length is different (less). Furthermore, the content of EXCLUDE_H must calculate according to the new gradation. Using the fixed octilinear length instead

of ordinary horizontal can obtain the optimal answer, because the number of hyperedges is decided by the length. If the length is minimal, it has less chance to overlap others.

Chapter 6 Conclusion and Future Work

This thesis is focused on hypergraphs with one backbone. Adding the constraint backbone can make the hypergraph cleaner. Moreover, some optimization problems which cannot be solved may have solution if each hypergraph has only backbone.

In the aspect of crossing minimization, problems can have solution in polynomial time with dynamic programming technique.

In the aspect of total length minimization, many problems can be transformed to the bipartite matching which can be solved in polynomial time.

Lastly, in the aspect of number of hyperedges, we use greedy method to solve a variety of related problems. It is also polynomial time.

This thesis is a first step towards hypergraph drawing whose hyperedge has one backbone. There are several future works: it may combine with the layout which has cluster relations, it may consider multiple backbones in each gap of other aesthetic criteria, it also can use curves instead of straight-line.

The first direction means maybe the points in the hypergraph can have different size, and maybe points that have relations among them must be put together. In these situations, how do the hypergraph be drawn prettier?

The second direction is about multiple backbones in each gap. In this thesis, we did not mention that in the minimization of total length, the problem must cannot be solved by the classic bipartite matching problem.

The final direction is huge, the hypergraph drawing can not just be orthogonal or octilinear. The only constraint is that every hyperedge has only one backbone. How about a hypergraph whose hyperedge is a curve?

In addition to theoretical level, implementation and evaluation of hypergraph drawing with backbones are also a direction of future work.

Bibliography

[1] Evmorfia N. Argyriou, Michael A. Bekos, Antonios Symvonis. Maximizing the Total Resolution of Graphs. 18th Graph Drawing International Symposium, pages 62-67, 2010.

[2] Michael A. Bekos, Sabine Cornelsen, Martin Fink, Seokhee Hong, Michael Kaufmann, Martin N¨ollenburg, Ignaz Rutter, and Antonios Symvonis. Many-to-One Boundary Labeling with Backbones. 21st Graph Drawing International Symposium, pages 244-255, 2013.

[3] Michael A. Bekos, Michael Kaufmann, Michael Kaufmann, Antonios Symvonis. Boundary Labeling with Octilinear Leaders. Journal of Algorithmica, vol. 57, pages 436–461, 2010.

[4] Francois Bertault and Peter Eades. Drawing Hypergraphs in the Subset Standard. In Proceedings of 8th Graph Drawing International Symposium, pages 164-169, 2000.

[5] Michael Baur and Ulrik Brandes. Crossing Reduction in Circular Layouts. 30th Graph-Theoretic Concepts in Computer Science International Workshop, pages 332-343, 2004.

[6] Walter Didimo, Peter Eades, Giuseppe Liotta. Drawing Graphs with Right Angle Crossings. In Proceedings of 11th Workshop on Algorithms and Data Structures, pages 206-217, 2009.

[7] Thomas Eschbach, Wolfgang G¨unther, Bernd Becker. Orthogonal Hypergraph Drawing for Improved Visibility. Journal of Graph Algorithms and Applications, vol. 10, pages 141-157, 2006.

[8] Thomas M. J. Fruchterman, Edward M. Reingold. Graph drawing by force-directed placement. Journal Software—Practice & Experience archive
Vol. 21, pages 1129 - 1164, 1991.

[9] Emden R. Gansner, Yehuda Koren. Improved Circular Layouts. 14th Graph Drawing International Symposium, pages 386-398, 2006.

[10] Martin Junghans. Visualization of Hyperedges in Fixed Graph Layouts. Master thesis of Brandenburg University of Technology Cottbus, 2008.

[11] Michael Kaufmann, Marc van Kreveld, and Bettina Speckmann. Subdivision Drawings of Hypergraphs. 16th Graph Drawing International Symposium, pages 396-407, 2008.

[12] Erkki Mäkinen. How to Draw a Hypergraph. International Journal of Computer Mathematics, vol. 34, pages 177-185, 1990.

[13] Sumio Masuda, Kazuo Nakajima, Toshinobu Kashiwabara, and Toshio fujisawa. Crossing Minimization in Linear Embeddings of Graphs. IEEE Transactions on Computers,vol. 39, pages 124–127, 1990.

[14] Georg Sander. Layout of Directed Hypergraphs with Orthogonal Hyperedges. 11th Graph Drawing International Symposium, pages 381-386, 2003.

[15] Sergey Pupyrev, Lev Nachmanson, Michael Kaufmann. Improving Layered Graph Layouts with Edge Bundling. 18th Graph Drawing International Symposium, pages 329-340, 2010.

[16] C. Walshaw. A Multilevel Algorithm for Force-Directed Graph Drawing. In Proceedings of 8th Graph Drawing International Symposium, pages 171-182, 2000.