



**HAL**  
open science

## Path-Based Supports for Hypergraphs

Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, Arnaud Sallaberry

► **To cite this version:**

Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, Arnaud Sallaberry. Path-Based Supports for Hypergraphs. *Journal of Discrete Algorithms*, 2012, 14, pp.248-261. 10.1016/j.jda.2011.12.009 . hal-00735894

**HAL Id: hal-00735894**

**<https://hal.science/hal-00735894>**

Submitted on 27 Mar 2013

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Path-Based Supports for Hypergraphs

Ulrik Brandes<sup>a</sup>, Sabine Cornelsen<sup>a</sup>, Barbara Pampel<sup>a</sup>, Arnaud Sallaberry<sup>b</sup>

<sup>a</sup>*Department of Computer and Information Science, University of Konstanz, Box 67, 78457 Konstanz, Germany*

<sup>b</sup>*CNRS UMR 5800 LaBRI, INRIA Bordeaux - Sud Ouest, Pkko, 351, cours de la Libération, 33405 Talence Cedex, France*

---

## Abstract

A path-based support of a hypergraph  $H$  is a graph with the same vertex set as  $H$  in which each hyperedge induces a Hamiltonian subgraph. While it is  $\mathcal{NP}$ -hard to decide whether a path-based support has a monotone drawing, to determine a path-based support with the minimum number of edges, or to decide whether there is a planar path-based support, we show that a path-based tree support can be computed in polynomial time if it exists.

*Keywords:* graph algorithm, graph drawing, hypergraph, metro map layout

---

## 1. Introduction

A *hypergraph* is a pair  $H = (V, A)$  where  $V$  is a finite set and  $A$  is a (multi-)set of non-empty subsets of  $V$ . The elements of  $V$  are called *vertices* and the elements of  $A$  are called *hyperedges*. A *support* (or *host graph*) of a hypergraph  $H = (V, A)$  is a graph  $G = (V, E)$  such that each hyperedge of  $H$  induces a connected subgraph of  $G$ , i.e., such that the graph  $G[h] := (h, \{e \in E, e \subseteq h\})$  is connected for every  $h \in A$ . See Fig. 1(b) for an example.

Applications for supports of hypergraphs are, e.g., in hypergraph coloring [2, 3], databases [4], or hypergraph drawing [5, 6, 7, 8]. E.g., see Fig. 1 for an application of a support for designing Euler diagrams. An *Euler diagram* of a hypergraph  $H = (V, A)$  is a drawing of  $H$  in the plane in which the vertices are drawn as points and each hyperedge  $h \in A$  is drawn as a simple closed region containing the points representing the vertices in  $h$  and not the points representing the vertices in  $V \setminus h$ . There are various well-formedness conditions for Euler diagrams, see e.g. [9, 8].

Recently, many papers have been devoted to the problem of deciding which classes of hypergraphs admit what kind of supports. It can be tested in linear time whether a hypergraph has a support that is a tree [10], a path or a cycle [7]. It can be decided in polynomial time whether a hypergraph has a tree support with bounded degrees [7] or a cactus support [11]. A minimum weighted tree support can be computed in polynomial time [12]. It is  $\mathcal{NP}$ -complete to decide whether a hypergraph has a planar support [5], a compact support [5, 6] or a 2-outerplanar support [7]. A support with the minimum

---

*Email addresses:* [Ulrik.Brandes@uni-konstanz.de](mailto:Ulrik.Brandes@uni-konstanz.de) (Ulrik Brandes), [Sabine.Cornelsen@uni-konstanz.de](mailto:Sabine.Cornelsen@uni-konstanz.de) (Sabine Cornelsen), [Barbara.Pampel@uni-konstanz.de](mailto:Barbara.Pampel@uni-konstanz.de) (Barbara Pampel), [arnaud.sallaberry@labri.fr](mailto:arnaud.sallaberry@labri.fr) (Arnaud Sallaberry)

A preliminary version of this paper was presented at the 21st International Workshop on Combinatorial Algorithms (IWOCA 2010) and appears in the corresponding proceedings [1].

Corresponding author phone: +49 7531 88 4375, fax: +49 7531 88 3577.

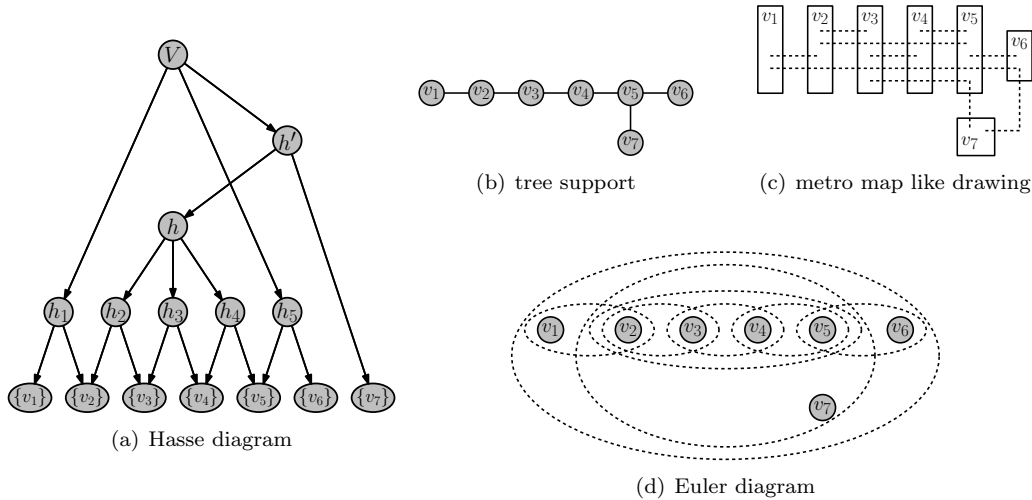


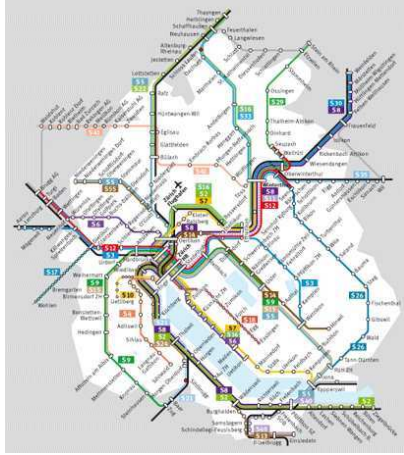
Figure 1: Three representations of the hypergraph  $H = (V, A)$  with hyperedges  $h_1 = \{v_1, v_2\}$ ,  $h_2 = \{v_2, v_3\}$ ,  $h_3 = \{v_3, v_4\}$ ,  $h_4 = \{v_4, v_5\}$ ,  $h_5 = \{v_5, v_6\}$ ,  $h = \{v_2, v_3, v_4, v_5\}$ ,  $h' = \{v_2, v_3, v_4, v_5, v_7\}$ , and  $V = \{v_1, \dots, v_7\}$ .

30 number of edges can be computed in polynomial time if the hypergraph is closed under  
 31 intersections [7]. If the set of hyperedges is closed under intersections and differences, it  
 32 can be decided in polynomial time whether the hypergraph has a planar or outerplanar  
 33 support [11].

34 In this paper we consider a restriction on the subgraphs of a support that are induced  
 35 by the hyperedges. A support  $G$  of a hypergraph  $H = (V, A)$  is called *path-based* if the  
 36 subgraph  $G[h]$  contains a *Hamiltonian path* for each hyperedge  $h \in A$ , i.e.,  $G[h]$  contains a  
 37 path that contains each vertex of  $h$ . This definition was motivated by the aesthetics of  
 38 metro map layouts. I.e., the hyperedges could be visualized as lines along the Hamiltonian  
 39 path in the induced subgraph of the support like the metro lines in a metro map. See  
 40 Fig. 2 for examples of metro maps, Fig. 3 for an example of natural sciences drawn in the  
 41 metro map anthology, and Fig. 1(c) and 6(f) for a representation of some hyperedges in  
 42 such a metro map like drawing. For metro map layout algorithms see, e.g., [13, 14].

43 We briefly consider monotone, planar, and minimum path-based supports. Our main  
 44 result is a characterization of those hypergraphs that have a path-based tree support  
 45 and a polynomial time algorithm for constructing path-based tree supports if they exist.  
 46 E.g., Fig. 1 shows an example of a hypergraph  $H = (V, A)$  that has a tree support but  
 47 no path-based tree support. However, the tree support in Fig. 1(b) is a path-based tree  
 48 support for  $(V, A \setminus \{V\})$ .

49 The contribution of this paper is as follows. In Section 2 we give the necessary defini-  
 50 tions. We then briefly discuss monotone path-based supports in Section 3 and mention  
 51 that finding a minimum path-based support or deciding whether there is a planar path-  
 52 based support, respectively, is  $\mathcal{NP}$ -hard. We consider path-based tree supports in Sect. 4.  
 53 In Section 4.1 we review a method for computing tree supports using the Hasse diagram.  
 54 In Section 4.2 we show how to apply this method to test whether a hypergraph has a  
 55 path-based tree support and if so how to compute one in polynomial time. Finally, in  
 56 Section 4.3 we discuss the run time of our method.



(a) local trains of Zurich



(b) metro of Amsterdam

Figure 2: Local train map of Zurich ([www.zvv.ch](http://www.zvv.ch)) and the metro map of Amsterdam ([www.amsterdam.info](http://www.amsterdam.info)). In (b) the union of all lines forms a tree.

57 **2. Preliminaries**

58 In this section, we give the necessary definitions that were not already given in the  
 59 introduction. Throughout this paper let  $H = (V, A)$  be a hypergraph. We denote by  
 60  $n = |V|$  the number of vertices,  $m = |A|$  the number of hyperedges, and  $N = \sum_{h \in A} |h|$   
 61 the sum of the sizes of all hyperedges of a hypergraph  $H$ . The *size of the hypergraph*  $H$   
 62 is then  $N + n + m$ . A hypergraph is a *graph* if all hyperedges contain exactly two vertices.  
 63 A hypergraph  $H = (V, A)$  is *closed under intersections* if  $h_1 \cap h_2 \in A \cup \{\emptyset\}$  for  $h_1, h_2 \in A$ .  
 64 We say that two hyperedges  $h_1, h_2$  *overlap* if  $h_1 \cap h_2 \neq \emptyset$ ,  $h_1 \not\subseteq h_2$ , and  $h_2 \not\subseteq h_1$ . A  
 65 hypergraph  $H = (V, A)$  is *connected* if for any pair of vertices  $v, w \in V$  there is a sequence  
 66 of hyperedges  $h_1, \dots, h_\ell \in A$  such that  $v \in h_1, w \in h_\ell$ , and  $h_i \cap h_{i+1} \neq \emptyset, i = 1, \dots, \ell - 1$ .

67 The *Hasse diagram* of a hypergraph  $H = (V, A)$  is the directed acyclic graph with  
 68 vertex set  $A \cup \{\{v\}; v \in V\}$  and there is an edge  $(h_1, h_2)$  if and only if  $h_2 \subsetneq h_1$  and there  
 69 is no set  $h \in A$  with  $h_2 \subsetneq h \subsetneq h_1$ . Fig. 1(a) shows an example of a Hasse diagram. Let  
 70  $(v, w)$  be an edge of a directed acyclic graph. Then we say that  $w$  is a *child* of  $v$  and  $v$   
 71 a *parent* of  $w$ . For a *descendant*  $d$  of  $v$  there is a directed path from  $v$  to  $d$  while for an  
 72 *ancestor*  $a$  of  $v$  there is a directed path from  $a$  to  $v$ . A *source* does not have any parents,  
 73 a *sink* no children and an *inner vertex* has at least one parent and one child.

74 **3. Path-Based Supports**

75 In a *metro map like drawing* of a hypergraph vertices are drawn as disjoint simple  
 76 closed regions in the plane and each hyperedge  $h$  is drawn as a curve  $\mathcal{C}_h$  with the end  
 77 points within the regions of different vertices of  $h$ , visiting the region of every vertex of  
 78  $h$  exactly once, not visiting the vertices not in  $h$ , and such that the pieces of  $\mathcal{C}_h$  within  
 79 the region of a vertex or between two such regions are simple. A *path-based support* of a  
 80 hypergraph  $H = (V, A)$  is a graph  $G$  such that  $G[h]$  contains a spanning path for every  
 81 hyperedge  $h \in A$ .

82 On one hand, a metro map like drawing of a hypergraph  $H = (V, A)$  induces a path-  
 83 based support  $G = (V, E)$  of  $H$ : For a hyperedge  $h \in A$  let  $p_h : v_1, \dots, v_{|h|}$  be the

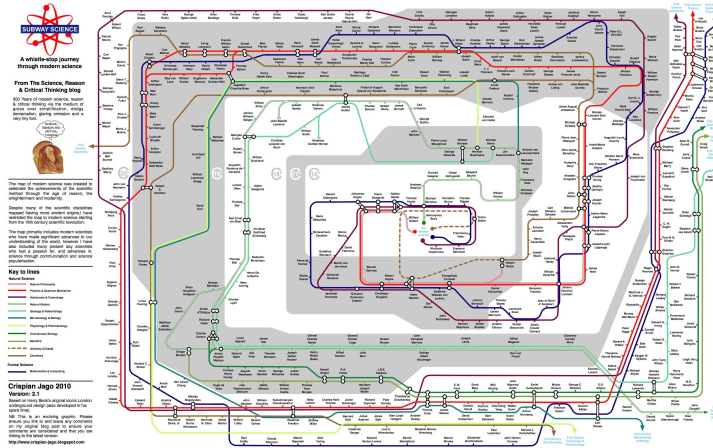


Figure 3: A map of modern science ([www.crispian.net](http://www.crispian.net)).

84 sequence of vertices of  $h$  in the order in which they are visited by the curve representing  
 85  $h$ . Starting with an empty set  $E$  add for every hyperedge  $h \in A$  with  $p_h : v_1, \dots, v_{|h|}$  the  
 86 edges  $\{v_{i-1}, v_i\}, i = 2, \dots, |h|$  to  $E$ . On the other hand, if we have a path-based support  
 87  $G$  of  $H$  and we fix for every hyperedge  $h \in A$  a spanning path  $p_h$  of  $G[h]$  then this induces  
 88 a metro map like drawing of  $H$ .

89 In order to have a readable metro map like drawing of a hypergraph it is typically  
 90 desirable to draw any curve representing a hyperedge without self intersection or even  
 91 monotone.

### 92 3.1. Monotone Path-Based Supports

93 A *drawing* of a graph is a mapping of each vertex to a distinct point in the plane and  
 94 of each edge to a simple curve between the image of its adjacent vertices not containing  
 95 the image of any other vertex. In a straight-line drawing of a graph each edge is drawn  
 96 as a line segment. Given a drawing of  $G$ , a path  $p$  of  $G$  is *monotone* with respect to  
 97 a straight line  $\ell$  – called the *axis of monotonicity* – if every line perpendicular to  $\ell$  intersects  
 98 the drawing of  $p$  in at most one point. Note that a path  $p$  in a straight-line drawing is  
 99 monotone with respect to the axis  $\ell$  if and only if the orthogonal projections of the vertices  
 100 of  $p$  on  $\ell$  appear along  $\ell$  in the order induced by  $p$ .

101 Let  $G = (V, E)$  be a path-based support of a hypergraph  $H = (V, A)$ . A drawing of  
 102  $G$  is *monotone* with respect to  $H$  if for each hyperedge  $h \in A$  there is a spanning path  
 103  $p_h$  of  $G[h]$  and a straight line  $\ell_h$  such that  $p_h$  is monotone with respect to the axis  $\ell_h$ .  $G$   
 104 is a *monotone* path-based support of  $H$  if  $G$  has a monotone drawing with respect to  $H$ .

105 **Remark 1.** *If  $G$  has a monotone drawing with respect to a hypergraph  $H$  then  $G$  has*  
 106 *a straight-line drawing that is monotone with respect to  $H$  with the same axes of mono-*  
 107 *tonicity.*

108 **PROOF.** Let a drawing  $\mathcal{D}$  of  $G$  that is monotone with respect to  $H = (V, A)$  be given  
 109 and let  $p_h, h \in A$  be a spanning path of  $G[h]$  that is monotone with respect to the axis  
 110  $\ell_h$ . If for each edge  $\{v, w\}$  of  $G$  the line segment between  $v$  and  $w$  does not contain any  
 111 vertex of  $G$  other than  $v$  or  $w$  then the straight-line drawing of  $G$  in which the vertices  
 112 are mapped to the same points as in  $\mathcal{D}$  is monotone with respect to  $H$ .

113 Consider now for two vertices  $v, w$  in a hyperedge  $h$  the distances  $\text{dist}_h(v, w)$  between  
 114 the orthogonal projections of  $v$  and  $w$  to  $\ell_h$ . Let  $\Delta$  be the minimum of all distances  
 115  $\text{dist}_h(v, w)$  over all  $h \in A$  and  $v, w \in h$  with  $v \neq w$ . Let  $0 < \varepsilon \leq \Delta/3$ .

116 Consider now the vertices of  $V$  in an arbitrary order  $v_1, \dots, v_n, n = |V|$ . For  $k =$   
 117  $1, \dots, n$ , we can now place  $v_k$  on the circle with radius  $\varepsilon$  around the position of  $v_k$  in  $\mathcal{D}$   
 118 but not on the intersection with the line through the already fixed drawings of  $v_i$  and  $v_j$ ,  
 119  $1 \leq i < j < k$ . The corresponding straight-line drawing is monotone with respect to  $H$   
 120 with the axes of monotonicity  $\ell_h, h \in A$ .  $\square$

121 **Remark 2.** *Not every path based support of a hypergraph is monotone.*

122 **PROOF.** Consider the following hypergraph. Let  $I = \{(i, j, k, \ell); 1 \leq i < j \leq 5, 1 \leq k <$   
 123  $\ell \leq 5, i < k, \{i, j\} \cap \{k, \ell\} = \emptyset\}$  be an index set representing unordered pairs of disjoint  
 124 edges of the complete graph  $K_5$ . Let  $V_I = \{v_i; i = 1, \dots, 5\} \cup \{v_{i,j,k,\ell,x}; (i, j, k, \ell) \in$   
 125  $I, x = 1, \dots, 3\}$ , let  $h_{ijkl} = \{v_i, v_{i,j,k,\ell,1}, v_j, v_{i,j,k,\ell,2}, v_k, v_{i,j,k,\ell,3}, v_\ell\}$ ,  $(i, j, k, \ell) \in I$ , let  
 126  $A_I = \{h_{ijkl}; (i, j, k, \ell) \in I\}$ , and let  $H_I = (V_I, A_I)$ . Let  $E$  contain the edges  $\{v_i, v_{i,j,k,\ell,1}\}$ ,  
 127  $\{v_{i,j,k,\ell,1}, v_j\}$ ,  $\{v_j, v_{i,j,k,\ell,2}\}$ ,  $\{v_{i,j,k,\ell,2}, v_k\}$ ,  $\{v_k, v_{i,j,k,\ell,3}\}$ ,  $\{v_{i,j,k,\ell,3}, v_\ell\}$  for  $(i, j, k, \ell) \in I$ .  
 128 The resulting path-based support  $G = (V, E)$  of  $H_I$  is shown in Fig. 4(a). Note that  
 129  $G[h_{ijkl}]$  is a path for any hyperedge  $h_{ijkl} \in A$  visiting the vertices  $v_i, v_j, v_k, v_\ell$  in this  
 130 order. Consider now any drawing of  $G$ . Since a  $K_5$  is not planar, there are two straight  
 131 line segments  $\overline{v_i v_j}, \overline{v_k v_\ell}, (i, j, k, \ell) \in I$  that intersect. Hence, the path  $G[h_{ijkl}]$  cannot be  
 132 drawn monotonously.  $\square$

133 **Remark 3.** *Every hypergraph has a monotone path-based support.*

134 **PROOF.** Order the vertices of  $H = (V, A)$  with respect to an arbitrary ordering  $<$ . The  
 135 support  $G_{<} = (V, E_{<})$  of  $H$  with respect to the ordering  $<$  is constructed as follows.  
 136 For each hyperedge  $\{v_1, \dots, v_k\} \in A$  with  $v_1 < \dots < v_k$  the edge set  $E_{<}$  contains the  
 137 edges  $\{v_{i-1}, v_i\}, i = 1, \dots, k$ . Assume now that in a drawing of  $G_{<}$  the x-value of a  
 138 vertex  $v$  is smaller than the x-value of the vertex  $w$  if  $v < w$  and that the edges are  
 139 drawn monotonously in x-direction. Then for each hyperedge  $h = \{v_1, \dots, v_k\} \in A$  with  
 140  $v_1 < \dots < v_k$  the path  $p_h : v_1, \dots, v_k$  is drawn monotonously with respect to the x-axis.  
 141 See Fig 4(c) for an example.  $\square$

142 Note that the problem of deciding whether a given support is a support with respect  
 143 to an ordering and if so, finding such an ordering, is closely related to the betweenness  
 144 problem [15].

145 **Theorem 1.** *Given a support  $G$  of a hypergraph  $H$*

- 146 1. *it is  $\mathcal{NP}$ -hard to decide whether  $G$  is a monotone path-based support of  $H$ , and*
- 147 2. *it is  $\mathcal{NP}$ -complete to decide whether there exists an ordering  $<$  of the vertex set*  
 148 *such that  $G$  is the support of  $H$  with respect to  $<$ ,*

149 *even if  $G$  has the minimum number of edges among all supports of  $H$ .*

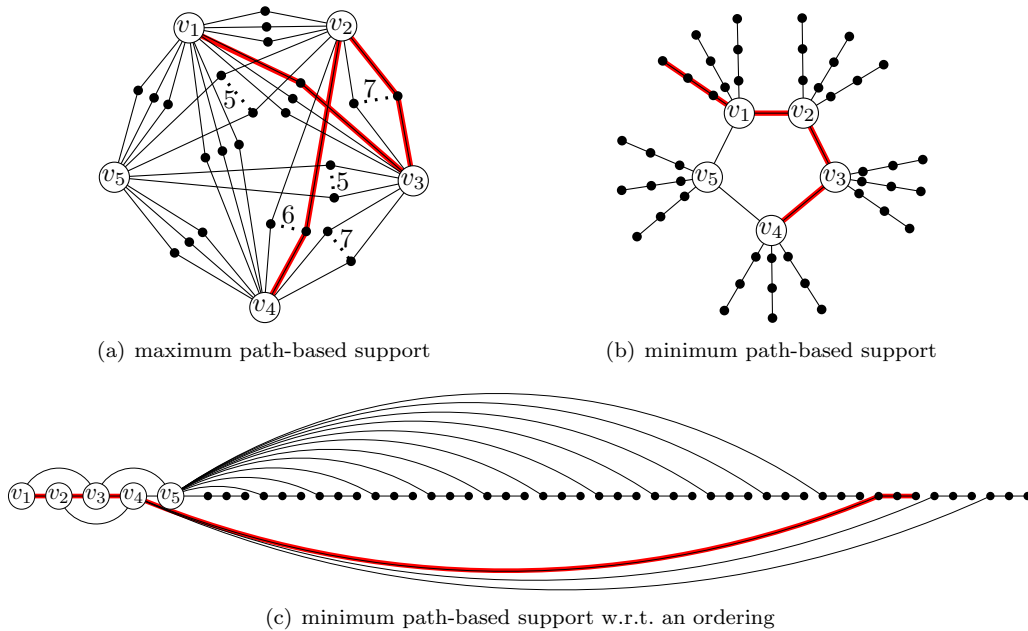


Figure 4: Three different supports for the hypergraph  $H_I$  introduced in Section 3.1. The small black vertices are the vertices  $v_{\sigma,x}, \sigma \in I, x = 1, 2, 3$ . The thick red path indicates the hyperedge  $h_{1324}$ .

150 PROOF.

151 1. Consider an instance of the *strictly monotone trajectory drawing problem* consisting of  
 152 a set of paths  $P$  on a set of vertices  $V_t$ . It is  $\mathcal{NP}$ -hard to decide whether the  
 153 vertices can be mapped to points in the plane such that each path is monotone with  
 154 respect to some axis (one for each path) [16].

155 Consider the hypergraph  $H = (V, A)$  with  $V$  containing  $V_t$  and for each path  $p \in P$   
 156 and each edge  $e \in p$  a vertex  $v_{ep}$ . The set  $A$  contains for each path  $p \in P$  a hyperedge  
 157  $h_p = \bigcup_{\{v,w\} \in p} \{v, v_{\{v,w\}p}, w\}$  as well as the hyperedges  $\{v, v_{\{v,w\}p}\}$  and  $\{v_{\{v,w\}p}, w\}$   
 158 for each edge  $\{v, w\} \in p$ . The graph  $G = (V, E)$  with  $E = \bigcup_{p \in P} \bigcup_{e \in p} \{\{v, v_{ep}\}; v \in e\}$   
 159 is a path-based support of  $H$  and has the minimum number of edges among all  
 160 supports of  $H$ .  $G$  is monotone if and only if  $P$  is drawable with each path monotone  
 161 with respect to some axis.

162 2. Consider an instance of the *betweenness problem* consisting of a set of vertices  $V_b$   
 163 and a set of constraints  $C$ . Each constraint  $c \in C$  consists of a sequence of three  
 164 vertices. It is  $\mathcal{NP}$ -complete to decide whether the vertices can be totally ordered  
 165 such that for each constraint  $c = (u, v, w)$  the vertex  $v$  is between the vertices  $u$  and  
 166  $w$  [15].

167 Consider the hypergraph  $H = (V, A)$  with  $V$  containing  $V_b$  and for each constraint  
 168  $c \in C$  vertices  $v_{c2}$  and  $v_{c4}$ . The set  $A$  contains for each  $c = (v_{c1}, v_{c3}, v_{c5}) \in C$  a  
 169 hyperedge  $h_c = \{v_{c1}, \dots, v_{c5}\}$  and hyperedges  $h_{ci} = \{v_{ci}, v_{c(i+1)}\}$  for  $1 \leq i \leq 4$ .  
 170 The graph  $G = (V, E)$  with  $E = \bigcup_{c \in C} \{h_{ci}; 1 \leq i \leq 4\}$  is a path-based support of  
 171  $H$  and has the minimum number of edges among all supports of  $H$ .

172 There is an ordering  $<$  of  $V$  such that  $G$  is the support of  $H$  with respect to  $<$   
 173 if and only if for each constraint  $c = (v_{c1}, v_{c3}, v_{c5}) \in C$  the five vertices in  $h_c$  are  
 174 either ordered  $v_{c1} < v_{c2} < v_{c3} < v_{c4} < v_{c5}$  or  $v_{c5} < v_{c4} < v_{c3} < v_{c2} < v_{c1}$ . Since

175 the vertices  $v_{c2}$  and  $v_{c4}$  do not appear in a hyperedge  $h_{c'}$  for any constraint  $c \neq c'$   
 176 it follows that there is an ordering  $<$  of  $V$  such that  $G$  is the support of  $H$  with  
 177 respect to  $<$  if and only if  $V_b$  can be totally ordered while satisfying all betweenness  
 178 constraints in  $C$ .  $\square$

### 179 3.2. Minimum Path-Based Supports

180 Assuming that each hyperedge contains at least one vertex, each hypergraph  $H =$   
 181  $(V, A)$  has a monotone path-based support  $G = (V, E)$  with at most  $N - m$  edges. Just  
 182 take the support  $G_{<}$  with respect to an arbitrary ordering  $<$  of the vertex set  $V$ . It is,  
 183 however,  $\mathcal{NP}$ -hard to find an ordering that minimizes the number of edges among all  
 184 path-based supports of  $H$  with respect to an ordering of the vertex set [17].

185 Further, note that a path-based support that minimizes the number of edges among  
 186 all path-based support of a hypergraph  $H$  with respect to some ordering of the vertex  
 187 set might not be a path-based support of  $H$  with the minimum number of edges over all  
 188 path-based supports of  $H$ . E.g., consider the hypergraph  $H_I$  from the previous section  
 189 (Fig. 4) or the hypergraph  $H$  with hyperedges  $\{1, 2, 4\}$ ,  $\{1, 3, 4\}$ , and  $\{2, 3, 4\}$  for an  
 190 easier example: the unique minimum path-based support of  $H$  is a star centered at 4  
 191 which cannot be created from any ordering of the vertex set. The problem of finding a  
 192 minimum path-based support remains, however,  $\mathcal{NP}$ -hard.

193 **Theorem 2.** *It is  $\mathcal{NP}$ -hard to minimize the number of edges among all path-based sup-*  
 194 *ports (or among all monotone path-based supports) of a hypergraph – even if the hyper-*  
 195 *graph is closed under intersections.*

196 **PROOF.** Reduction from Hamiltonian path. Let  $G = (V, E)$  be a graph. Let  $H =$   
 197  $(V, E \cup \{V\} \cup \{\{v\}; v \in V\})$  and  $K = |E|$ . Note that any support of  $H$  contains  $G$  as a  
 198 subgraph. Hence,  $H$  has a path-based support with at most  $K$  edges if and only if  $G$  is a  
 199 path-based support of  $H$  which is true if and only if  $G$  contains a Hamiltonian path.  $\square$

### 200 3.3. Planar Path-Based Supports

201 A graph is *planar* if it has a drawing in which no pair of edges intersect but in common  
 202 end points. For the application of Euler diagram like drawings, planar supports are of  
 203 special interest. However, like for general planar supports, the problem of testing whether  
 204 there is a path-based planar support is hard.

205 **Theorem 3.** *It is  $\mathcal{NP}$ -complete to decide whether a hypergraph – even if it is closed*  
 206 *under intersections – has a path-based planar support.*

207 **PROOF.** The support that Johnson and Pollak [5] constructed to prove that it is  $\mathcal{NP}$ -  
 208 complete to decide whether there is a planar support, was already path-based.  $\square$

## 209 4. Path-Based Tree Supports

210 In this section we show how to decide in polynomial time whether a given hypergraph  
 211 has a path-based tree support. If such a support exists, it is at the same time a path-based  
 212 support of minimum size, a monotone path-based support [18], and a planar path-based  
 213 support. Moreover the intersection of any subset of hyperedges induces again a path in a  
 214 path-based tree support. So far it is known how to decide in linear time whether there is  
 215 a path-based tree support if  $V \in A$  [7].



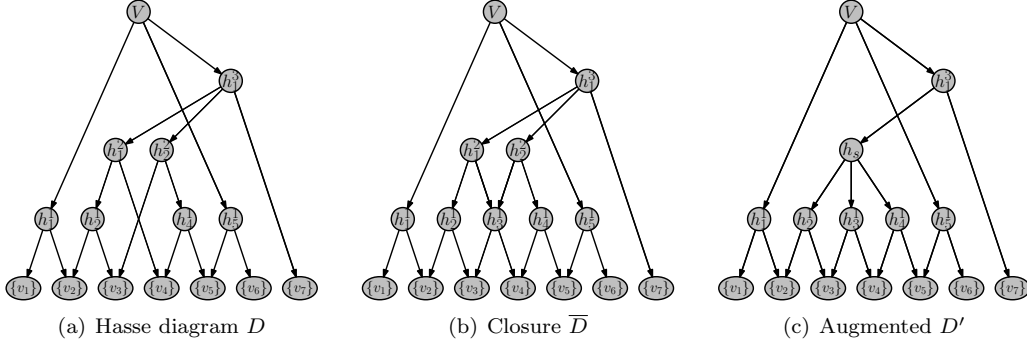


Figure 5: Illustration of the augmented Hasse Diagram for the hypergraph  $H = (V, A)$  indicated in 5(a).  $\bar{A} = A \cup \{h_3^1, \{v_2\}, \dots, \{v_5\}\}$ . The two hyperedges  $h_1^2$  and  $h_2^2$  are both implied but no summary edges. They are not present in the augmented Hasse diagram. The summary hyperedge  $h_s$  is added to  $A'$ .

216 4.1. Constructing a Tree Support from the Hasse Diagram

217 A support with the minimum number of edges and, hence, a tree support if one exists  
 218 can easily be constructed from the Hasse diagram if the hypergraph is closed under inter-  
 219 sections [7]. Note, however, that the number of intersections of any subset of hyperedges  
 220 could be exponential in the size of the hypergraph.

221 To construct a tree support of an arbitrary hypergraph  $H = (V, A)$ , it suffices to  
 222 consider the *augmented Hasse diagram* – a representation of “necessary” intersections of  
 223 hyperedges. The definition is as follows. First consider the smallest set  $\bar{A}$  of subsets of  $V$   
 224 that contains  $A$  and that is closed under intersections. Consider the Hasse diagram  $\bar{D}$  of  
 225  $\bar{H} = (V, \bar{A})$ . Note that any tree support of  $H$  is also a tree support of  $\bar{H}$ : The intersection  
 226 of two subtrees is again a subtree.

227 Let  $h_1, \dots, h_k$  be the children of a hyperedge  $h$  in  $\bar{D}$ . The hyperedge  $h \in \bar{A}$  is *implied*  
 228 if the hypergraph  $(h_1 \cup \dots \cup h_k, \{h_1, \dots, h_k\})$  is connected and *non-implied* otherwise. Let  
 229  $\{h_1, \dots, h_k\}$  be a maximal subset of the children of a non-implied hyperedge in  $\bar{A}$  such  
 230 that  $(h_1 \cup \dots \cup h_k, \{h_1, \dots, h_k\})$  is connected. Then  $h_1 \cup \dots \cup h_k$  is a *summary hyperedge*.  
 231 Note that a summary hyperedge might not be in  $\bar{A}$ . Let  $A'$  be the set of subsets of  $V$   
 232 containing the summary hyperedges, the hyperedges in  $\bar{A}$  that are not implied, and the  
 233 sources of  $\bar{D}$ . For an example consider Fig. 5(c). In this example, the hyperedge  $h_s$  is a  
 234 summary hyperedge,  $h_1^3$  and  $h_1^1, \dots, h_5^1$  are non-implied, and  $V$  is a source.

235 The augmented Hasse diagram of  $H$  is the Hasse diagram  $D'$  of  $H' = (V, A')$ . If  
 236  $H$  has a tree support, then the augmented Hasse diagram has  $\mathcal{O}(n + m)$  vertices and  
 237 can be constructed in  $\mathcal{O}(n^3 m)$  time [7] (without explicitly constructing the closure under  
 238 intersection  $\bar{A}$ ). Further note that if  $H$  has a tree support and  $h \in A'$  is non-implied,  
 239 then all children of  $h$  in  $D'$  are disjoint: Otherwise there would be a summary hyperedge  
 240 between  $h$  and intersecting children.

241 If a tree support  $G = (V, E)$  of  $H$  exists, it can be constructed as follows [7]. Starting  
 242 with an empty graph  $G$ , we proceed from the sinks to the sources of  $D'$ . If  $h \in A'$   
 243 is not implied, choose an arbitrary ordering  $h_1, \dots, h_k$  of the children of  $h$  in  $D'$ . We  
 244 assume that at this stage,  $G[h_i], i = 1, \dots, k$  are already connected subgraphs of  $G$ . For  
 245  $j = 2, \dots, k$ , choose vertices  $v_j \in \bigcup_{i=1}^{j-1} h_i$ ,  $w_j \in h_j$  and add edges  $\{v_j, w_j\}$  to  $E$ .

246 If we want to construct a path-based tree support, then  $G[h_j], j = 1, \dots, k$  are paths  
 247 and as vertices  $v_{j+1}$  and  $w_j$  for the edges connecting  $G[h_j]$  to the other paths, we choose

248 the end vertices of  $G[h_j]$ . The only choices that remain are the ordering of the children  
 249 of  $h$  and the choice of which end vertex of  $G[h_j]$  is  $w_j$  and which one is  $v_{j+1}$ . The implied  
 250 hyperedges give restrictions on how these choices might be done.

251 *4.2. Choosing the Connections: A Characterization*

252 When we want to apply the general method introduced in Sect. 4.1 to construct a  
 253 path-based tree support  $G$ , we need to make sure that we do not create vertices of degree  
 254 greater than 2 in  $G[h]$  when processing non-implied hyperedges contained in an implied  
 255 hyperedge  $h$ .

256 **Definiton 1 (Conflicting Hyperedges).** *Two overlapping hyperedges  $h', h'' \in A'$  have*  
 257 *a conflict if there is some hyperedge in  $A'$  that contains both  $h'$  and  $h''$ . Two overlapping*  
 258 *hyperedges  $h', h'' \in A'$  have a conflict with respect to  $h \in A'$  if  $h'$  has a conflict with  $h''$ ,*  
 259  *$h' \cap h'' \subseteq h$  and  $h$  is a child of  $h'$  or  $h''$ . In that case we say that  $h'$  and  $h''$  are conflicting*  
 260 *hyperedges of  $h$ . Let  $A'_h$  be the set of conflicting hyperedges of  $h$ . Let  $A_h^c$  be the set of*  
 261 *children  $h_i$  of  $h$  such that  $h \in A'_{h_i}$ .*

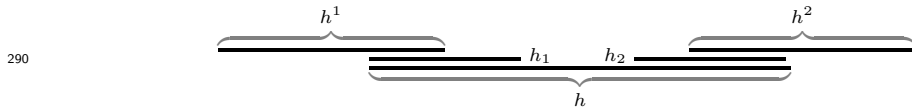
262 E.g., consider the hypergraph in Fig. 5(c). Then  $h_1^3$  and  $h_1^1$  are both contained in the  
 263 hyperedge  $V$  and they both contain  $\{v_2\}$ . Hence, they have a conflict. Further, is the  
 264 intersection  $h_1^3 \cap h_1^1 = \{v_2\}$  contained in the child  $h_s$  of  $h_1^3$ . Hence,  $h_1^3$  has a conflict with  
 265  $h_1^1$  with respect to  $h_s$ . Similarly does  $h_1^3$  have a conflict with  $h_5^1$  with respect to  $h_s$  and  
 266 we have on one hand  $A'_{h_s} = \{h_1^1, h_2^1, h_1^3\}$ . On the other hand does  $h_s$  have a conflict with  
 267  $h_1^1$  with respect to  $h_2^1$  and with  $h_5^1$  with respect to  $h_4^1$  and we have  $A_{h_s}^c = \{h_2^1, h_4^1\}$ . Note  
 268 that there might be hyperedges that have a conflict but not with respect to any of their  
 269 children. As an example see the hyperedges  $h_1^4$  and  $h_2^4$  in Fig. 6(a). In the lemmas in this  
 270 section, we will prove that it suffices if the algorithm considers only conflicts with respect  
 271 to some child.

272 Assume now that  $H$  has a path-based tree support  $G$  and let  $h', h'' \in A'$  be such that  
 273  $h'$  and  $h''$  have a conflict with respect to a child  $h$  of  $h''$ . Since  $h' \cup h''$  is contained in a  
 274 hyperedge it follows that  $G[h' \cup h'']$  is the subgraph of a path. Since in addition  $h'$  and  $h''$   
 275 intersect and  $G[h']$  and  $G[h'']$  are paths, it follows that  $G[h' \cup h'']$  is also a path. Hence,  
 276 we have the following situation.



278 Note especially that among the two end vertices of  $G[h'']$  exactly one is contained in  $h'$   
 279 and that this end vertex is also an end vertex of  $G[h]$ . This yields the following three  
 280 types of restrictions on the connections of the paths.

- 281
- 282 1.  $G[h' \setminus h]$  and  $G[h'' \setminus h]$  must be paths that are attached to different end vertices of  
 283  $G[h]$ .
  - 284 2. Assume further that  $h'''$  does also have a conflict with  $h''$  with respect to  $h$ . Then  
 285 both,  $G[h' \setminus h]$  and  $G[h''' \setminus h]$ , must be appended to the common end vertex of  $G[h]$   
 and  $G[h'']$ .
  - 286 3. Assume further that  $h_2, h_1 \in A_h^c, h_2 \neq h_1$ . Let  $h^i \in A'_h$  have a conflict with  $h$   
 287 with respect to  $h_i, i = 1, 2$ , respectively. Then  $G[h^i \setminus h]$  has to be appended to  
 288 the common end vertex of  $G[h]$  and  $G[h_i]$ . Hence,  $G[h^1 \setminus h]$  and  $G[h^2 \setminus h]$  must be  
 289 appended to different and vertices of  $G[h]$ .



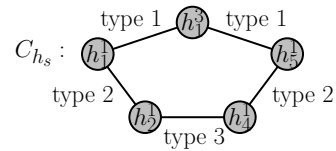
291 E.g., consider the hypergraph  $H = (V, A)$  in Fig. 5(c). Then on one hand,  $h_1^3$  has a conflict  
 292 with  $h_1^1$  and  $h_5^1$  with respect to  $h_s$ . Hence, by the first type of restrictions  $G[h_1^1 \setminus h_s]$  and  
 293  $G[h_5^1 \setminus h_s]$  must be appended to the same end vertex of  $G[h_s]$ , i.e. the end vertex of  $G[h_s]$   
 294 to which  $G[h_1^3 \setminus h_s]$  is not appended. On the other hand,  $h_1^1$  and  $h_s$  have a conflict with  
 295 respect to  $h_2^1$ , while  $h_5^1$  and  $h_s$  have a conflict with respect to  $h_4^1$ . Hence, by the third  
 296 type of restrictions it follows that  $G[h_1^1 \setminus h_s]$  and  $G[h_5^1 \setminus h_s]$  must be appended to different  
 297 end vertices of  $G[h]$ . Hence, there is no path-based tree support for  $H$ .

298 This motivates the following definition of conflict graphs.

299 **Definiton 2 (Conflict Graph).** *The conflict graph  $C_h, h \in A'$  is a graph on the vertex*  
 300 *set  $A'_h \cup A_h^c$ . The conflict graph  $C_h$  contains the following three types of edges.*

- 301 1.  $\{h', h''\}, h', h'' \in A'_h$  if  $h'$  and  $h''$  have a conflict with respect to  $h$ .  
 302 2.  $\{h', h_1\}, h' \in A'_h, h_1 \in A_h^c$  if  $h' \in A'_h$  and  $h'$  and  $h$  have a conflict with respect to  
 303  $h_1$ .  
 304 3.  $\{h_1, h_2\}, h_1, h_2 \in A_h^c, h_1 \neq h_2$ .

E.g., consider the hypergraph  $H = (V, A)$  in Fig. 5(c).  
 Then the conflict graph  $C_{h_s}$  contains the edges  $\{h_1^3, h_5^1\}$   
 and  $\{h_1^3, h_1^1\}$  of type one, the edges  $\{h_2^1, h_1^1\}$  and  $\{h_4^1, h_5^1\}$   
 305 of type 2 and the edge  $\{h_2^1, h_4^1\}$  of type 3. (See the figure  
 on the right.) Hence,  $C_{h_s}$  contains a cycle of odd length,  
 reflecting that there is no suitable assignment of the end  
 vertices of  $G[h_s]$  to  $h_1^1, h_5^1$  and  $h_1^3$ .



306 **Theorem 4.** *A hypergraph  $H = (V, A)$  has a path-based tree support if and only if*

- 307 1.  $H$  has a tree support,  
 308 2. no hyperedge contains three pairwise overlapping hyperedges  $h_1, h_2, h_3 \in A'$  with  
 309  $h_1 \cap h_2 = h_2 \cap h_3 = h_1 \cap h_3$ , and  
 310 3. all conflict graphs  $C_h, h \in A', |h| > 1$  are bipartite.

311 From the observations before the definition of the conflict graph it is clear that the  
 312 conditions of Theorem 4 are necessary for a path-based tree support. In the remainder  
 313 of this section, we prove that the conditions are also sufficient.

314 In the following assume that the conditions of Theorem 4 are fulfilled. We show  
 315 in Algorithm 1 how to construct a path-based tree support  $G$  of  $H$ . We consider the  
 316 vertices of the augmented Hasse diagram  $D'$  from the sinks to the sources in a *reversed*  
 317 *topological order*, i.e., we consider a hyperedge only if all its children in  $D'$  have already  
 318 been considered. During the algorithm, a conflicting hyperedge  $h'$  of a hyperedge  $h$  is  
 319 labeled with the end vertex  $v$  of  $G[h]$  if the path  $G[h' \setminus h]$  will be appended to  $v$ . We  
 320 will call this label  $\text{side}_h(h')$ . Concerning the choice of the ordering of the children in  
 321 Line 8 of Algorithm 1: the sets  $A_h^c, h \in A'$  contain at most two hyperedges – otherwise  
 322 the subgraph of  $C_h$  induced by  $A_h^c$  contains a triangle and, hence, is not bipartite.

323 Algorithm 1 constructs a tree support  $G$  of  $H$  [7]. Before we show that  $G$  is a path-  
 324 based tree support, we illustrate the algorithm with an example. Consider the hypergraph  
 325  $H$  in Fig. 6. We show how the algorithm proceeds  $h_1^5$  and all its descendants in  $D'$ . For



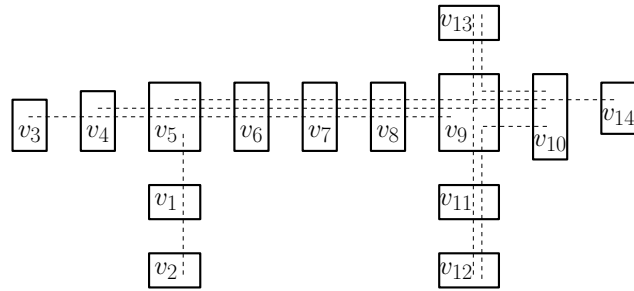
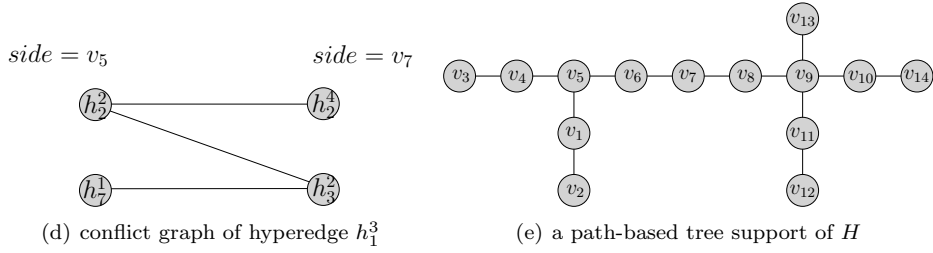
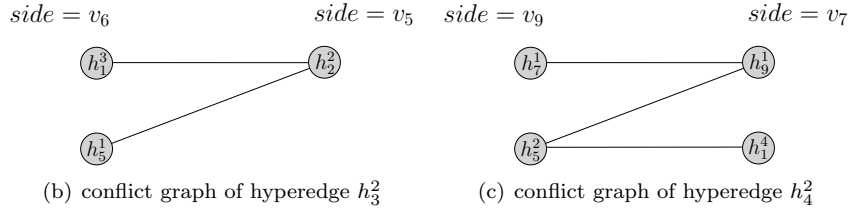
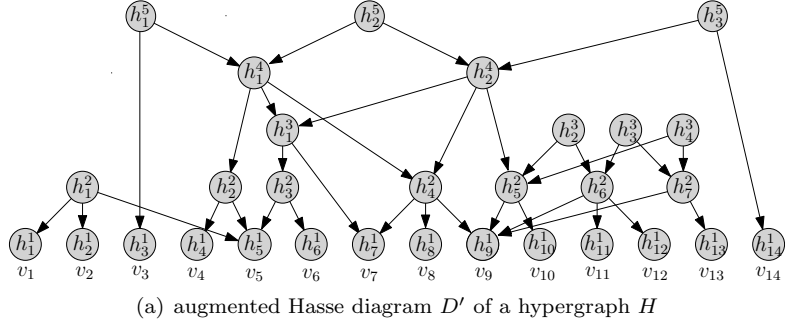


Figure 6: Illustration of Algorithm 1.

326 the hyperedges  $h_3^1, h_4^1, h_6^1$ , and  $h_8^1$  the conflict graphs are empty. For the other leaves we  
 327 have

$$\begin{aligned} \text{side}_{h_5^1}(h_2^2) = \text{side}_{h_5^1}(h_3^2) = \text{side}_{h_5^1}(h_1^3) = \text{side}_{h_5^1}(h_2^4) &= v_5, \\ \text{side}_{h_7^1}(h_4^2) = \text{side}_{h_7^1}(h_1^3) &= v_7, \text{ and} \\ \text{side}_{h_9^1}(h_4^2) = \text{side}_{h_9^1}(h_1^4) = \text{side}_{h_9^1}(h_5^2) = \text{side}_{h_9^1}(h_6^2) = \text{side}_{h_9^1}(h_7^2) &= v_9. \end{aligned}$$

328 When operating  $h_2^2$  and  $h_3^2$ , respectively, we add edges  $\{v_4, v_5\}$  and  $\{v_5, v_6\}$ , respec-  
 329 tively, to  $G$ . While the conflict graph of  $h_2^2$  does only contain  $h_5^1$  with  $\text{side}_{h_2^2}(h_5^1) = v_4$ , the  
 330 assignment of side in  $C_{h_3^2}$  is illustrated in Fig. 6(b).  $h_4^2$  has a conflict with respect to the  
 331 children  $h_7^1$  and  $h_9^1$ . Hence, we add edges  $\{v_7, v_8\}$  and  $\{v_8, v_9\}$  to  $G$ . The conflict graph  
 332 of  $h_4^2$  is shown in Fig. 6(c). When operating  $h_1^3$  we can choose  $h_1 = h_3^2$  and  $h_2 = h_7^1$ , since  
 333  $\text{side}_{h_3^2}(h_1^3) = v_6$  and  $\text{side}_{h_7^1}(h_1^3) = v_7$ . We add the edge  $\{v_6, v_7\}$  to  $G$ . The conflict graph  
 334  $C_{h_1^3}$  is shown in Fig. 6(d). The hyperedge  $h_1^4$  is implied and we set  $\text{side}_{h_1^4}(h_4^2) = v_4$ . We  
 335 can finally connect  $v_3$  to  $v_4$  or  $v_9$  when operating  $h_1^5$ .

336 To prove the correctness of Algorithm 1, it remains to show that all hyperedges of  $H$   
 337 induce a path in  $G$ . Since we included all inclusion maximal hyperedges of  $H$  in  $A'$ , it  
 338 suffices to show this property for all hyperedges in  $A'$ . We start with a technical lemma.

339 **Lemma 5.** *Let  $h'$  and  $h''$  be two overlapping hyperedges and let  $h'$  be not implied. Then*  
 340 *there is a hyperedge  $h \in A'$  with  $h' \cap h'' \subseteq h \subsetneq h'$ .*

341 **PROOF.** Let  $h_c \in \overline{A}$  be maximal with  $h' \cap h'' \subseteq h_c \subsetneq h'$ . The hyperedge  $h_c$  is a child of  
 342 the non-implied hyperedge  $h'$  in  $\overline{D}$ . Consider the summary hyperedge  $h$  with  $h_c \subseteq h \subsetneq h'$ .  
 343 By definition of  $A'$  it follows that  $h \in A'$ .  $\square$

344 For an edge  $\{v, w\}$  of  $G$  let  $h_{vw}$  be the intersection of all hyperedges of  $A'$  that contain  
 345  $v$  and  $w$ . Note that  $h_{vw}$  is not implied since  $v$  and  $w$  are contained in different children  
 346 of  $h_{vw}$  in  $\overline{D}$  and  $\{v, w\}$  is an edge of the tree support  $G$  of  $\overline{H}$ . Hence,  $h_{vw} \in A'$ .

347 **Lemma 6.** *Let Conditions 1-3 of Theorem 4 be fulfilled and let  $G = (V, E)$  be the graph*  
 348 *computed in Algorithm 1. Let  $h', h'' \in A'$  have a conflict with respect to a child  $h$  of  $h'$*   
 349 *and let  $G[h']$  and  $G[h'']$  be paths. Then*

- 350 1.  $\text{side}_g(h'') = \text{side}_h(h'')$  for all  $g \in A'$  with  $h' \cap h'' \subseteq g \subseteq h$ ,
- 351 2.  $\text{side}_h(h'') \in h''$ ,
- 352 3.  $\text{side}_h(h'')$  is an end vertex of  $G[h']$ ,
- 353 4.  $G[h' \setminus h'']$  is a path, and
- 354 5.  $\text{side}_h(h'')$  is adjacent in  $G$  to a vertex of  $h'' \setminus h'$ .

355 **PROOF.** We prove the lemma by induction on the sum of the steps in which  $h'$  and  $h''$   
 356 were considered in Algorithm 1. If  $h'$  and  $h''$  had been considered in the first two steps,  
 357 then at least one of them is a leaf of  $D'$  and, hence,  $h'$  and  $h''$  have no conflict. So there  
 358 is nothing to show. Let now  $h'$  and  $h''$  be considered in later steps. Let  $h'' \in A'$  have a  
 359 conflict with  $h'$  with respect to a child  $h$  of  $h'$  and let  $G[h']$  and  $G[h'']$  be paths.

360 **1. + 2. if  $h' \cap h'' \in A'$ :** There is nothing to show if  $h = h' \cap h''$ . So let  $h_1$  be the child  
 361 of  $h$  with  $h_1 \supseteq h' \cap h''$ . Then  $h, h''$  have a conflict with respect to  $h_1$ . Hence,  
 362  $C_h$  contains the path  $h', h'', h_1$ . By the inductive hypothesis on Property 3, it



389 Let  $h'_1, h''_1 \in A'$  be minimal with  $h' \supseteq h'_1 \supseteq h' \cap h''$  and  $h'' \supseteq h''_1 \supseteq h' \cap h''$ ,  
 390 respectively. Assume first that  $h' \cap h'' \in A'$ . Then  $C_{h' \cap h''}$  contains the triangle  
 391  $h_{vw}, h'_1, h''_1, h_{vw}$  and, hence, is not bipartite.

392 Assume now that  $h' \cap h'' \notin A'$ . By the already proven part of Property 5 it follows  
 393 that there is an edge  $\{v, x\}$  of  $G$  with  $x \in h'_1 \setminus h$ . We have  $h_c = h_{vw} \cap h'' \supseteq h_{vw} \cap h_{vx}$ .  
 394 Further, the child of  $h_{vx}$  that contains  $h_{vx} \cap h$  equals  $h_{vx} \cap h$ . Since  $h'_1$  is implied  
 395 and  $h_{vx}$  not, it follows that  $h'_1 \neq h_{vx}$  and, hence,  $h_{vx} \not\supseteq h' \cap h''$ . Hence, either  
 396  $h_{vx} \cap h \subseteq h_{vw} \cap h$  or  $h_{vw} \cap h \subsetneq h_{vx} \cap h \subsetneq h' \cap h''$ . In the first case let  $h_1 \in A'$   
 397 be minimal with  $h_{vw} \cap h \subsetneq h_1 \subseteq h$ . Then there is the triangle  $h_{vw}, h_{vx}, h_1, h_{vw}$  in  
 398  $C_{h \cap h_{vw}}$ . In the latter case let  $h_1 \in A'$  be minimal with  $h_{vx} \cap h \subsetneq h_1 \subseteq h$ . Then  
 399 there is the triangle  $h_{vw}, h_{vx}, h_1, h_{vw}$  in  $C_{h \cap h_{vx}}$ .

400 **4.:** By the inductive hypothesis  $G[h \setminus h'']$  is a path. Further,  $h$  and  $h'$  share  $\text{side}_h(h'') \in h''$   
 401 as a common end vertex. By the precondition of the lemma,  $G[h']$  is a path. Hence,  
 402  $G[h' \setminus h'']$  is a path.

403 **5. if  $h' \cap h'' \in A'$ :** If  $h \neq h' \cap h''$ , let  $h_1$  be the child of  $h$  with  $h' \cap h'' \subseteq h_1$ . By the  
 404 inductive hypothesis  $\text{side}_{h_1}(h'')$  is adjacent in  $G$  to a vertex of  $h'' \setminus h = h'' \setminus h'$  and  
 405 by Property 1,  $\text{side}_{h_1}(h'') = \text{side}_h(h'')$ .

406 If  $h = h' \cap h''$ , let  $h'_1 \in A'$  be minimal with  $h \subsetneq h'_1 \subseteq h''$ . Applying Property 3 with  
 407  $h'_1$  as “ $h'$ ” and  $h'$  as “ $h''$ ” reveals that  $\text{side}_h(h')$  is an end vertex of  $G[h'_1]$ . Since  
 408  $G[h'_1]$  is a path, it follows that some vertex of  $h'_1 \setminus h$  is adjacent to  $\text{side}_h(h'')$ .  $\square$

409 **Lemma 7.** *If Conditions 1-3 of Theorem 4 are fulfilled, then all hyperedges in  $A'$  induce*  
 410 *a path in the graph  $G$  constructed in Algorithm 1.*

411 **PROOF.** Again, we prove the lemma by induction on the step in which  $h$  was considered  
 412 in Algorithm 1. There is nothing to show if  $h$  had been considered in the first step. So  
 413 assume that  $h \in A'$  and that  $G[h]$  contains a vertex  $v$  of degree greater than two.

414 Let  $u_1, u_2, u_3$  be the first three vertices connected to  $v$  in  $G$ . Let  $h_i = h_{vu_i}, i = 1, 2, 3$ .  
 415 Then  $h_1, h_2, h_3$  are all three contained in  $h$  and its intersection contains  $v$ . Hence, any  
 416 two of them have a conflict if and only if one of them is not contained in the other. A  
 417 case distinction reveals that we wouldn't have appended all three,  $u_1, u_2$  and  $u_3$ , to  $v$ .

418  $h_2 = h_3$ : Since  $h_3$  contains no vertex of degree higher than two, it follows that  $u_1 \notin h_3$ ,  
 419  $h_3 \cap h_1 = \{v\}$ . Hence,  $h_1$  and  $h_3$  have a conflict with respect to the common child  
 420  $\{v\}$ , contradicting that  $v$  is added in the middle of  $h_3$ .

421  $h_1 = h_2$  **or**  $h_1 = h_3$ : These cases are analogous to the first case.

422  $h_1 \subsetneq h_3$ : Like in the first case it follows that  $u_2 \notin h_3$ . Let  $h'_i, i = 2, 3$  be the child of  
 423  $h_i$  that contains  $v$ . Then  $h_2$  and  $h_3$  have a conflict with respect to  $h'_i, i = 2, 3$ .  
 424 Since we add the edge  $\{v, u_i\}$  to  $G$  when we process  $h_i$ , it follows on one hand that  
 425  $\text{side}_{h'_i}(h_i) = v$ . On the other hand, since  $h_1$  is contained in  $h_3$  and  $v \in h_1$ , it follows  
 426 that  $h_1 \subseteq h'_3$ . Hence,  $h'_3$  has more than one vertex. If  $h'_3 \neq h_3 \cap h_2$ , then  $v$  is the only  
 427 end vertex of  $G[h'_3]$  that is contained in  $h_2$ . By Lemma 6 Property 2 it follows that  
 428  $\text{side}_{h'_3}(h_2) = v$  and hence,  $\text{side}_{h'_3}(h_3) \neq v$ . If  $h'_3 = h_3 \cap h_2$ , let  $v' \neq v$  be the other  
 429 end vertex of  $h'_2$ . Since we know that  $\text{side}_{h'_2}(h_2) = v$ , it follows that  $\text{side}_{h'_2}(h_3) = v'$ .  
 430 Hence, by Lemma 6 Property 1 we can conclude that  $\text{side}_{h'_3}(h_3) = v'$ . In both cases  
 431 we have a contradiction.





---

**Algorithm 2:** Conflict Computation.

---

**Input** : augmented Hasse diagram  $D'$  of a hypergraph; vertex  $h$

**Output:** for each child  $h_c$  of  $h$  the vertices  $h'$  with  $\text{LABEL}(h') = \text{CONFLICT}(h_c)$

**Data** : there are the following vertex labels

$\text{LABEL}(h') = \text{ANC}$  iff  $h \subsetneq h'$

$\text{LABEL}(h') = \text{NOT-ANC}$  only if  $h \cup h'$  not contained in any source of  $D'$

$\text{LABEL}(h') = \text{DESC}(h_c)$  iff  $h' \subseteq h_c$  for exactly one child  $h_c$  of  $h$

$\text{LABEL}(h') = \text{MULTI-DESC}$  iff  $h'$  is contained in more than one child of  $h$

$\text{LABEL}(h') = \text{NOT-CONFLICT}$  only if  $h \cap h'$  not contained in any child of  $h$   
and  $h \cup h'$  contained in some source of  $D'$

$\text{LABEL}(h') = \text{CONFLICT}(h_c)$  only if  $h_c \cap h' \neq \emptyset$  for a child  $h_c$  of  $h$   
and  $h \cup h'$  contained in some source of  $D'$

ANCESTOR(vertex  $h'$ ) **begin**

**foreach** parent  $h''$  of  $h'$  **do**

$\text{LABEL}(h'') \leftarrow \text{ANC}$ ;

    ANCESTOR( $h''$ );

**end**

DESCENDANT(vertex  $h'$ , vertex  $h_c$ ) **begin**

**if**  $\text{LABEL}(h') = \text{DESC}(h'_c), h_c \neq h'_c$  **then**

$\text{LABEL}(h') \leftarrow \text{MULTI-DESC}$ ;

**else**

$\text{LABEL}(h') \leftarrow \text{DESC}(h_c)$ ;

**foreach** child  $h''$  of  $h'$  **do**

**if**  $\text{LABEL}(h'') \neq \text{MULTI-DESC}$  **then**

      DESCENDANT( $h'', h_c$ );

**end**

UP-SEARCH(vertex  $h'$ , vertex  $h_c$ ) **begin**

**foreach** parent  $h''$  of  $h'$  **do**

**if**  $\text{LABEL}(h'') \in \{\emptyset, \text{CONFLICT}(h'_c), h'_c \neq h_c\}$  **then**

      UP-SEARCH( $h'', h_c$ );

**if**  $\text{LABEL}(h') = \text{CONFLICT}(h'_c), h_c \neq h'_c$  **then**

$\text{LABEL}(h') \leftarrow \text{NOT-CONFLICT}$ ;

**else if**  $\text{LABEL}(h') \neq \text{DESC}(h_c)$  **then**

**if**  $\text{LABEL}(h'') \in \{\text{CONFLICT}(h_c), \text{ANC}, \text{NOT-CONFLICT}\}$  **then**

$\text{LABEL}(h') \leftarrow \text{CONFLICT}(h_c)$ ;

**if**  $\text{LABEL}(h') \neq \text{CONFLICT}(h_c)$  **then**

$\text{LABEL}(h') \leftarrow \text{NOT-ANC}$ ;

**end**

**begin**

  clear all labels;

$\text{LABEL}(h) \leftarrow \text{NOT-CONFLICT}$ ;

  ANCESTOR( $h$ );

**foreach** child  $h_c$  of  $h$  **do**

    DESCENDANT( $h_c, h_c$ );

**foreach** vertex  $h_d$  of  $D'$  with  $\text{LABEL}(h_d) \in \{\text{DESC}(h_c); h_c \text{ child of } h\}$  **do**

    UP-SEARCH( $h_d, h_c$ );

**end**

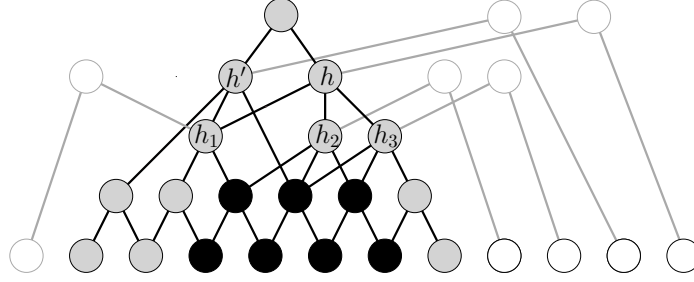


Figure 9:  $h'$  and  $h$  do not have a conflict with respect to any child of  $h$  but the label of  $h'$  is  $\text{CONFLICT}(h_1)$  in the end of Algorithm 2 applied to  $h$ . All descendants of  $h_2$  (black vertices) are labeled  $\text{MULTI-DESC}$ .

463 **Lemma 9.** *Let  $D'$  be the augmented Hasse diagram of a hypergraph that has a path-based*  
 464 *tree support and let  $h'$  and  $h$  have a conflict with respect to a child  $h_c$  of  $h$ . Then the label*  
 465 *of  $h'$  in the end of Algorithm 2 applied to  $D'$  and  $h$  is  $\text{CONFLICT}(h_c)$ .*

466 **PROOF.** Let  $G$  be a path-based tree support of a hypergraph and let  $h'$  and  $h$  have a  
 467 conflict with respect to a child  $h_c$  of  $h$ .

- 468 1. Let  $v$  be the end vertex of  $G[h]$  that is contained in  $h'$ . Then  $v$  and all its ancestors  
 469 on the path from  $\{v\}$  to  $h_c$  are labeled  $\text{DESC}(h_c)$  (and not  $\text{MULTI-DESC}$ ).
- 470 2. If there was a descendant of  $h'$  labeled  $\text{DESC}(h'_c)$  for a child  $h'_c \neq h_c$  of  $h$ , then  $h'$   
 471 contains a vertex of  $h$  that is not contained in  $h_c$ . Hence,  $h_c$  does not contain  $h \cap h'$ ,  
 472 contradicting that  $h$  and  $h'$  have a conflict with respect to  $h_c$ . Hence, Algorithm 2  
 473 does not label  $h'$  with  $\text{NOT-CONFLICT}$ .

474 For a path  $P$  in  $D'$  from  $h'$  to  $v$  let  $h_P$  be the first vertex on  $P$  that is labeled  $\text{DESC}(h_c)$ .  
 475 Assume that among all such vertices  $h_P$  is the one to which the procedure  $\text{UP-SEARCH}$  of  
 476 Algorithm 2 is applied first. Then  $\text{UP-SEARCH}(h_P, h_c)$  labels  $h'$  with  $\text{CONFLICT}(h_c)$ .  $\square$

477 Note, however, that the converse of Lemma 9 is not true. More precisely, if  $h'$  is labeled  
 478  $\text{CONFLICT}(h_c)$  in the end of Algorithm 2 applied to  $h$  then indeed do  $h$  and  $h'$  have a  
 479 conflict, but  $h_c$  does not have to contain  $h \cap h'$ . The reason for this is that all descendants  
 480 of  $h'$  that are no descendants of  $h_c$  are descendants of several children of  $h$  and, hence,  
 481 labeled  $\text{MULTI-DESC}$ . Fig. 9 shows an example.

482 **Theorem 10.** *It can be tested in  $\mathcal{O}(n^3m)$  time whether a hypergraph has a path-based*  
 483 *tree support and if so, such a support can be constructed within the same time bounds.*

484 **PROOF.** Let  $H$  be a hypergraph. First test in linear time whether there is a tree support  
 485 for  $H$  [10]. Let  $D'$  be the augmented Hasse diagram of  $H$ . The method works in four  
 486 steps.

- 487 1. Start with an empty array  $\text{conflict}$  indexed with pairs of inner vertices of  $D'$ . Set  
 488  $\text{conflict}_{h,h'} \leftarrow h_c$  if and only if  $h'$  is labeled  $\text{CONFLICT}(h_c)$  in the end of Algorithm 2  
 489 applied to  $D'$  and  $h$ .
- 490 2. For each pair  $h, h'$  of inner vertices of  $D'$ , test whether  $\text{conflict}_{h,h'}$  contains  $h \cap h'$ .  
 491 Otherwise set  $\text{conflict}_{h,h'} \leftarrow \emptyset$ . Now, if  $H$  has a path-based tree support, then  $h, h'$   
 492 has a conflict with respect to the child  $h_c$  of  $h$  if and only if  $h_c = \text{conflict}_{h,h'}$ .
- 493 3. Apply Algorithm 1 to compute a support  $G$ . If the algorithm stops without com-  
 494 puting a support, then  $H$  does not have a path-based tree support.

495 4. Test whether every hyperedge induces a path in  $G$ . If not,  $H$  does not have a  
496 path-based tree support.

497  $D'$  has  $\mathcal{O}(n + m)$  vertices,  $\mathcal{O}(n^2 + nm)$  edges, and can be computed in  $\mathcal{O}(n^3m)$  time if  
498  $H$  has a tree support [7]. Algorithm 2 visits every edge of  $D'$  at most twice and, hence,  
499 runs in  $\mathcal{O}(n^2 + nm)$  time for each of the  $\mathcal{O}(n)$  inner vertices of  $D'$ .

500 We may assume that the hyperedges are given as sorted lists of their elements. If not  
501 given in advance, these lists could straight forwardly be computed from  $D'$  in  $\mathcal{O}(n^3 + mn^2)$   
502 time by doing a graph search from each leaf. Now, for each of the  $\mathcal{O}(n^2)$  pairs  $h, h'$  of  
503 inner vertices, it can be tested in  $\mathcal{O}(n)$  time whether  $\text{conflict}_{h,h'}$  contains  $h \cap h'$ .

504 The sum of the sizes of all conflict graphs is in  $\mathcal{O}(n^2)$ . Hence, Algorithm 1 runs in  
505  $\mathcal{O}(n^2 + mn)$  time. For each of the  $\mathcal{O}(m)$  hyperedges  $h$ , it can be tested in  $\mathcal{O}(n)$  time  
506 whether  $G[h]$  is a path. Hence, the overall run time is dominated by the computation of  
507 the augmented Hasse diagram and is in  $\mathcal{O}(n^3m)$ .  $\square$

## 508 5. Conclusion

509 We have introduced path-based supports for hypergraphs. Hence, as a new model,  
510 we considered a restriction on the appearance of those subgraphs of a support that are  
511 induced by the hyperedges. We have discussed that monotone path-based supports are  
512 desirable. We have shown that it is  $\mathcal{NP}$ -hard to decide whether a given path-based  
513 support is monotone or to find a path-based support with the minimum number of edges.  
514 Further, it is  $\mathcal{NP}$ -complete to decide whether there is a planar path-based support. As a  
515 main result, we characterized those hypergraphs that have a path-based tree support and  
516 we gave an algorithm that computes a path-based tree support in  $\mathcal{O}(n^3m)$  run time if it  
517 exists. Our algorithm completed the paths for the hyperedges in the order in which they  
518 appear in a reversed topological ordering of the augmented Hasse diagram. To connect  
519 these subpaths in the right order, we introduced a conflict graph for each hyperedge  $h$   
520 and colored its vertices with the end vertices of the path induced by  $h$ .

## 521 References

- 522 [1] U. Brandes, S. Cornelsen, B. Pampel, A. Sallaberry, Path-based supports for hyper-  
523 graphs, in: C. Iliopoulos, W. Smyth (Eds.), Proceedings of the 21st International  
524 Workshop on Combinatorial Algorithms (IWOCA 2010), Vol. 6460 of Lecture Notes  
525 in Computer Science, Springer, 2011, pp. 20–33.
- 526 [2] D. Král', J. Kratochvíl, H.-J. Voss, Mixed hypercacti, Discrete Mathematics 286  
527 (2004) 99–113.
- 528 [3] C. Bujtás, Z. Tuza, Color-bounded hypergraphs, II: Interval hypergraphs and hyper-  
529 trees, Discrete Mathematics 309 (2009) 6391–6401.
- 530 [4] C. Beeri, R. Fagin, D. Maier, M. Yannakakis, On the desirability of acyclic database  
531 schemes, Journal of the Association for Computing Machinery 30 (4) (1983) 479–513.
- 532 [5] D. S. Johnson, H. O. Pollak, Hypergraph planarity and the complexity of drawing  
533 Venn diagrams, Journal of Graph Theory 11 (3) (1987) 309–325.

- 534 [6] M. Kaufmann, M. van Kreveld, B. Speckmann, Subdivision drawings of hypergraphs,  
535 in: I. G. Tollis, M. Patrignani (Eds.), Proceedings of the 16th International Symposi-  
536 um on Graph Drawing (GD 2008), Vol. 5417 of Lecture Notes in Computer Science,  
537 Springer, 2009, pp. 396–407.
- 538 [7] K. Buchin, M. van Kreveld, H. Meijer, B. Speckmann, K. Verbeek, On planar sup-  
539 ports for hypergraphs, in: D. Eppstein, E. R. Gansner (Eds.), Proceedings of the  
540 17th International Symposium on Graph Drawing (GD 2009), Vol. 5849 of Lecture  
541 Notes in Computer Science, Springer, 2010, pp. 345–356.
- 542 [8] P. Simonetto, D. Auber, D. Archambault, Fully automatic visualisation of overlap-  
543 ping sets, *Computer Graphics Forum* 28 (3) (2009) 967–974.
- 544 [9] J. Flower, A. Fish, J. Howse, Euler diagram generation, *Journal on Visual Languages*  
545 and Computing 19 (6) (2008) 675–694.
- 546 [10] R. E. Tarjan, M. Yannakakis, Simple linear-time algorithms to test chordality of  
547 graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs,  
548 *SIAM Journal on Computing* 13 (3) (1984) 566–579.
- 549 [11] U. Brandes, S. Cornelsen, B. Pampel, A. Sallaberry, Blocks of hypergraphs applied  
550 to hypergraphs and outerplanarity, in: C. Iliopoulos, W. Smyth (Eds.), Proceedings  
551 of the 21st International Workshop on Combinatorial Algorithms (IWOCOA 2010),  
552 Vol. 6460 of Lecture Notes in Computer Science, Springer, 2011, pp. 201–211.
- 553 [12] E. Korach, M. Stern, The clustering matroid and the optimal clustering tree, *Math-*  
554 *ematical Programming, Series B* 98 (2003) 385 – 414.
- 555 [13] M. Nöllenburg, An improved algorithm for the metro-line crossing minimization prob-  
556 lem, in: D. Eppstein, E. R. Gansner (Eds.), Proceedings of the 17th International  
557 Symposium on Graph Drawing (GD 2009), Vol. 5849 of Lecture Notes in Computer  
558 Science, Springer, 2010, pp. 381–392.
- 559 [14] A. Wolff, Drawing subway maps: A survey, *Informatik-Forschung und Entwicklung*  
560 22 (1970) 23–44.
- 561 [15] J. Opatrny, Total ordering problem, *SIAM Journal on Computing* 8 (1) (1979) 111–  
562 114.
- 563 [16] B. Pampel, Constrained graph drawing, Ph.D. thesis, University of Konstanz, to  
564 appear (2011).
- 565 [17] D. S. Johnson, S. Krishnan, J. Chhugani, S. Kumar, S. Venkatasubramanian, Com-  
566 pressing large boolean matrices using reordering techniques, in: M. A. Nascimento,  
567 M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, K. B. Schiefer (Eds.), Pro-  
568 ceedings of the 13th International Conference on Very Large Data Bases (VLDB '04),  
569 Morgan Kaufmann, 2004, pp. 13–23.
- 570 [18] P. Angelini, E. Colasante, G. Di Battista, F. Frati, M. Patrignani, Monotone draw-  
571 ings of graphs, in: U. Brandes, S. Cornelsen (Eds.), Proceedings of the 18th Inter-  
572 national Symposium on Graph Drawing (GD 2010), Vol. 6502 of Lecture Notes in  
573 Computer Science, Springer, 2011, pp. 13–24.