Journal of Computer Languages 71 (2022) 101136

Contents lists available at ScienceDirect

Journal of Computer Languages

journal homepage: www.elsevier.com/locate/cola

Minimising line segments in linear diagrams is NP-hard

Peter Chapman *, Kevin Sim, Huang Hao Chen

School of Computing, Edinburgh Napier University, UK

ARTICLE INFO

Keywords:
Linear diagrams
Complexity
Set-based visualisation

ABSTRACT

Linear diagrams have been shown to be an effective method of representing set-based data. Moreover, a number of guidelines have been proven to improve the efficacy of linear diagrams. One of these guidelines is to minimise the number of line segments appearing in a diagram. We show this problem to be NP-hard.

1. Introduction

Set-based data is prevalent in a number of domains, in particular ontologies [1] and their application to, for example medicine [2–4], and law [5]. These fields demand precision and accuracy with the interpretation of data, and thus approaches which aid the end-user with the interpretation are beneficial. Visualisation is one such approach: by representing information diagrammatically, it can be conveyed in a compact form [6].

Amongst the factors limiting the uptake of visualisations are *usability* [7], *scalability* [8], and *automated support* [9,10]. These factors are related: without adequate tool support, drawing diagrams that convey a large amount of data is difficult, but scalability is not solely reliant on automated procedures. Similarly, the scalability of a diagrammatic notation will affect its usability, but is not the sole contributing factor. In this paper, we address the third factor (automated support) directly, with the intention of ameliorating issues with the usability and scalability. For a given representation of set-based data, linear diagrams, we prove that satisfying a key drawing guideline is NP-hard. Through the reduction, we identify other areas which can provide effective drawing algorithms for linear diagrams.

This paper is structured as follows. In Section 2, we examine in detail the objects of study: linear diagrams, whilst in Section 3 we show the problem of minimising line segments is NP-hard. In Section 4 we provide a brief overview of approaches that are now available to solve draw linear diagrams in an effective manner.

2. Background and related work

Linear diagrams are the focus of this article. They were originally formulated by Leibniz [11], and use parallel (horizontal) lines to represent sets and their intersections. We give a brief overview of the syntax and semantics of linear diagrams pertinent to this paper here; the reader is directed to [12] for a more complete description.

Fig. 1 shows a linear diagram, consisting of six *sets*: each horizontal space (here coloured) represents a given set. The sets are composed of individual *line segments*. In Fig. 1, for example, there is one line segment representing the set *News*; meanwhile, four line segments represent the set *Android*. The vertical space in the diagram is divided into *overlaps*, which represent the intersections between the sets. An overlap can be encoded as a list of sets, precisely those which have a line segment in that overlap. For example, the right-most overlap of Fig. 1 may be encoded as $[Books, Stars, News]$.

In each overlap, the represented intersection is that which contains precisely the represented sets which have a line segment in that overlap. This example overlap then represents the set intersection:

$$Books \cap Stars \cap News \cap \overline{Android} \cap \overline{Cars} \cap \overline{Media}$$

where $\bar{A}$ represents the complement of the set A. We can now define a linear diagram:

Definition 1. A linear diagram $d = (S(d), o(d))$ is a pair consisting of a list of **sets** of d, $S(d)$, and a list of **overlaps** of d, $o(d)$. The **length of** d, denoted $|d|$, is given by the length of $o(d)$. The number of line segments of d is denoted $seg(d)$.

This definition uses lists, an ordered structure, to define linear diagrams. One could define the sets and overlaps of a linear diagram as sets, in a similar fashion to the curves and zones of Euler and Venn diagrams [13]. We would then have an abstract description of a linear diagram, which could be instantiated by different concrete (drawn) diagrams. For example, a different vertical ordering of the sets would keep the abstract description unchanged, but produce a different drawn diagram. This abstract–concrete distinction makes sense in the region-based diagrams. For linear diagrams, however, we can define both the underlying data, and the way in which it could be represented, at the same time. For the purposes of this paper, we will only be concerned

* Corresponding author.
E-mail addresses: p.chapman@napier.ac.uk (P. Chapman), k.sim@napier.ac.uk (K. Sim), 40506146@live.napier.ac.uk (H.H. Chen).

https://doi.org/10.1016/j.cola.2022.101136
Received 11 November 2021; Received in revised form 10 June 2022; Accepted 8 July 2022
Available online 18 July 2022

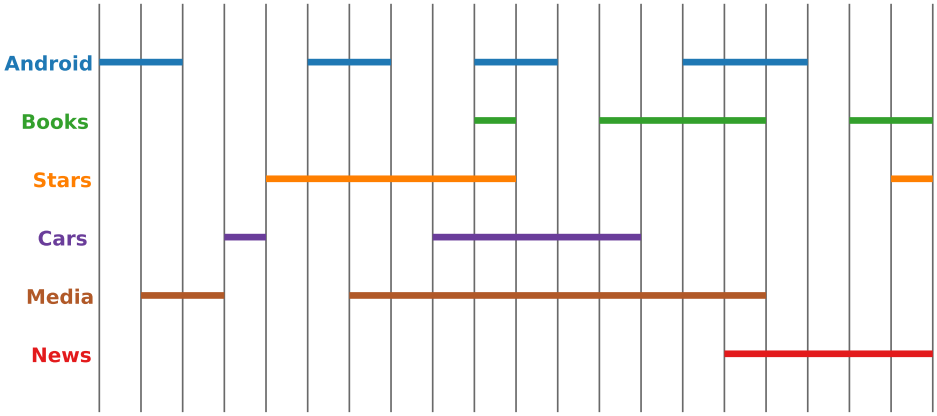

Fig. 1. Visualising sets: linear diagrams. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

with the ordering of the overlaps, and thus we require the following definition to equate two linear diagrams which represent the same collection of sets:

Definition 2. A pair of linear diagrams $d = (S(d), [o_1, \ldots, o_n])$ and $d_p = (S(d), [o'_1, \ldots, o'_n])$ are **equivalent** iff there exists a permutation p such that $p([o_1, \ldots, o_n]) = [o'_1, \ldots, o'_n]$. If d and d_p are equivalent, denoted $d =_{LD} d_p$, then we say that d is a **re-arrangement** of d_p (and vice versa).

This definition can easily be extended to include re-orderings of the list of sets, but is not necessary for this work.

Linear diagrams were shown to be more effective than Euler and Venn diagrams at representing set-based information in [14]. In [15], meanwhile, linear diagrams performed well against Euler diagrams when representing set-based information with cardinality, whilst in [16] linear diagrams performed at a similar level to mosaic diagrams. Drawing guidelines for linear diagrams were developed in [17], which can be seen implemented in Fig. 1: vertical grey guidelines are used to delineate one overlap from another; each set is drawn using a single colour; the line segments are narrow; and some others.

Chiefly of interest for this paper is that one should "*draw linear diagrams with a minimal number of line segments*" [17]. A drawing algorithm was given in that paper, but the question of whether more effective drawing algorithms existed (i.e. those which produced equivalent linear diagrams with fewer line segments) was not addressed. That a minimal number of line segments should be used is supported by other work. Whilst not the primary focus of [18], Stapleton et al. noted that "broken lines were problematic" for users. In [19], users were given the ability to minimise the number of line segments for particular sets of interest: those that did were significantly more accurate, confident and quick in answering questions relating to the sets of interest. Meanwhile, [20] described how as line segments increased, the perceived clutter of a diagram was found to increase as well. Perceived clutter is a measure which is found to correlate well (inversely) with efficacy in both Euler diagrams [21] and linear diagrams [22].

The goal of this paper is to attempt to provide solutions for the drawing guideline of [17], namely:

Problem 1. Given a linear diagram d, which re-arrangement of d contains the smallest number of line segments? More formally, given a linear diagram d, find $p^\star$ such that $seg(d_{p^\star}) = \min_p seg(d_p)$, for all p such that $d =_{LD} d_p$.

3. On complexity

Problem 1 is a combinatorial optimisation problem. It makes sense, then, to discuss the computational complexity of Problem 1. In this section, we show the Consecutive Block Minimisation Problem (CBMP) reduces to Problem 1. The CBMP which was shown to be NP-hard in [23]. Even a severely restricted version was shown to be NP-hard in [24]. We thus prove that Problem 1 is NP-hard.

> Consecutive Block Minimisation Problem: (As given in [24]) Given a binary matrix A (of size $m \times n$), let A_p be the matrix induced by a permutation p of the columns of A. The score of a permutation, $s(A_p)$, is the number of entries $a_{i,p(j)}$ such that:
>
> - $a_{i,p(j)} = 1$ **and**
> - either
>   - $a_{i,p(j+1)} = 0$, **or**
>   - $p(j) = n$.
>
> Which permutation p minimises $s(A_p)$?

We begin by giving the transformation:

Algorithm 1 Transforming CBMP to Problem 1

```
1:  procedure TRANSFORM(A)                          ▷ A an m × n binary matrix
2:      S(d) ← [s_1, ..., s_m]                      ▷ Create a list of m distinct set names
3:      o(d) ← []
4:      for i = 1, ..., n do
5:          o_i ← []
6:          for j = 1, ..., m do
7:              if a_{i,j} = 1 then
8:                  o_i ← o_i ++ [s_j]              ▷ ++ appends two lists
9:              end if
10:         end for
11:         o(d) ← o(d) ++ [o_i]
12:     end for
13:     d ← (S(d), o(d))
14: end procedure
```

As an example, consider the binary matrix:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Then, `transform(A)` could be:

$$\texttt{transform}\,(\texttt{A}) = ([s_1, s_2, s_3], [[s_1, s_3], [s_3], [s_1], [s_2]]).$$

Obviously, the choice of set names is arbitrary: they need only be a distinct set of names. This diagram would be drawn as shown in Fig. 2.

The complexity of `transform` is $O(mn)$ (where A has m rows and n columns), hence is polynomial. We next prove that the number of line segments in `transform`(A) is equal to $s(A)$.

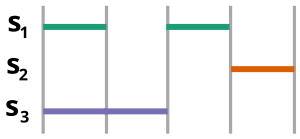

Fig. 2. Transforming a binary matrix.

Claim 1. *Given a binary matrix A of size $m \times n$:*

$$seg(transform(A)) = s(A)$$

Proof. Let A be given. We prove the claim by induction on n, the number of columns of A. We need only consider the case of a single row of A: both $s(A)$ and $seg(d)$ (for any d) can be calculated by counting the blocks (resp. segments) in each row (resp. set) separately, and summing the results. In light of this, let $i \in \{1, \dots, m\}$ be given. We denote $s_i(A)$ to be the block contribution of row i, and $seg_i(d)$ to be the segment contribution of the ith set in d.

When $n = 1$, either $a_{i,1} = 0$ or $a_{i,1} = 1$. In the former case, $s_i(A) = 0$ and in the latter $s_i(A) = 1$. Meanwhile, in the former case $\texttt{transform}(A) = ([\dots, t_i, \dots], [o])$, where $t_i \notin o$, and thus $seg_i(\texttt{transform}(A)) = 0 = s_i(A)$. In the latter case, $\texttt{transform}(A) = ([\dots, t_i, \dots], [[\dots, t_i, \dots]])$, and so there is one line segment for set t_i. Thus, we have

$$seg_i(\texttt{transform}(A)) = 1 = s_i(A).$$

Assume that the claim is true for $n = k$ (i.e. the induction hypothesis is that when $n = k$, $s_i(A) = seg_i(\texttt{transform}(A))$, and consider $n = k + 1$. There are four cases, depending on the values of $a_{i,k}$ and $a_{i,k+1}$. For readability, we will refer to the matrix formed from the first k columns of A by A':

1. $a_{i,k} = 0$, $a_{i,k+1} = 1$. Given the definition of s, we have $s_i(A) = s_i(A') + 1$. Similarly, there is one new line segment in $\texttt{transform}(A)$ in set t_i compared with $\texttt{transform}(A')$, and thus $seg_i(\texttt{transform}(A)) = seg_i(\texttt{transform}(A')) + 1$. Using the induction hypothesis, then $s_i(A) = seg_i(\texttt{transform}(A))$.

In the remaining cases, given the definition of $s(A)$, we have that $s_i(A) = s_i(A')$. Similarly, no new line segments for the ith set are introduced when moving from $\texttt{transform}(A')$ to $\texttt{transform}(A)$. Thus, since the induction hypothesis gives $s_i(A') = seg_i(\texttt{transform}(A'))$, we have that $s_i(A) = seg_i(\texttt{transform}(A))$, as required. The details are given below:

2. $a_{i,k} = 0$, $a_{i,k+1} = 0$. $s_i(A) = s_i(A')$ owing to the first clause in the description of s. Meanwhile, $t_i \notin o_k$ and $t_i \notin o_{k+1}$, and thus no new line segment is added.
3. $a_{i,k} = 1$, $a_{i,k+1} = 0$. $s_i(A) = s_i(A')$ owing to the second clause in the description of s: in the case of A', the last column contains a 1; and in the case of A, the penultimate column contains a 1, followed by a 0 in the last column. Meanwhile $t_i \in o_k$ but $t_i \notin o_{k+1}$, and thus no new line segment is added.
4. $a_{i,k} = 1$, $a_{i,k+1} = 1$. $s_i(A) = s_i(A')$ owing to the second clause in the description of s: in both A' and A, the last column contains a 1. Meanwhile, $t_i \in o_k$ and $t_i \in o_{k+1}$, and thus no new line segment is added.

Thus, for all n, $s(A) = seg(\texttt{transform}(A))$, as required. ■

We thus have that a permutation minimises $seg(\texttt{transform}(A_p))$ if and only if p also minimises $s(A_p)$. Then, as a consequence of $\texttt{transform}$ being a polynomial time transformation, we have that Problem 1 is at least as hard as CBMP, whence:

Theorem 1. *Problem 1 is NP-hard.*

4. Conclusions and future work

One of the findings of [17] was that "linear diagrams should be drawn with a minimal number of line segments". The work of this paper has shown that this is an NP-hard problem, and through the reduction from CBMP has demonstrated that any algorithm for minimising CBMP will also minimise line segments in a linear diagram. Consecutive block minimisation is an active field of research, with heuristic methods based on iterated local search [25], transformation to the travelling salesman problem (see e.g. [26]) and other closely related problems (e.g. minimising the number of 0s appearing as gaps [27]) all giving possible algorithms for drawing linear diagrams. Because the instances on which these other algorithms are tested are generally much larger than linear diagrams that appear in the literature, the scalability of drawing large diagrams can be addressed using these other approaches.

CRediT authorship contribution statement

Peter Chapman: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Kevin Sim:** Software, Resources, Formal analysis, Writing – review & editing. **Huang Hao Chen:** Software, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, Daniele Nardi, et al., The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.
[2] Joanne S. Luciano, Bosse Andersson, Colin Batchelor, Olivier Bodenreider, Tim Clark, Christine K. Denney, Christopher Domarew, Thomas Gambet, Lee Harland, Anja Jentzsch, et al., The translational medicine ontology and knowledge base: driving personalized medicine by bridging the gap between bench and bedside, in: Journal of Biomedical Semantics, Vol. 2, Springer, 2011, p. S1.
[3] Melissa A. Haendel, Christopher G. Chute, Peter N. Robinson, Classification, ontology, and precision medicine, N. Engl. J. Med. 379 (15) (2018) 1452–1462.
[4] Jean-Baptiste Lamy, Rosy Tsopra, Rainbio: Proportional visualization of large sets in biology, IEEE Trans. Vis. Comput. Graphics (2019).
[5] V. Richard Benjamins, Pompeu Casanovas, Joost Breuker, Aldo Gangemi, Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications, Vol. 3369, Springer, 2005.
[6] J. Larkin, H. Simon, Why a diagram is (sometimes) worth ten thousand words, J. Cogn. Sci. 11 (1987) 65–99.
[7] Peter Rodgers, A survey of Euler diagrams, J. Vis. Lang. Comput. 25 (3) (2014) 134–155.
[8] Michael Burch, Natalia Konevtsova, Julian Heinrich, Markus Hoeferlin, Daniel Weiskopf, Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study, IEEE Trans. Vis. Comput. Graphics 17 (12) (2011) 2440–2448.
[9] Arnaud Hubaux, Andreas Classen, Marcilio Mendonça, Patrick Heymans, A preliminary review on the application of feature diagrams in practice, VaMoS 10 (2010) 53–59.
[10] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, E. Giannopoulou, Ontology visualization methods - a survey, ACM Comput. Surv. 39 (4) (2007) 10.
[11] L. Couturat, Opuscules et fragments inédits de Leibniz, in: Felix Alcan, 1903.
[12] Peter Chapman, Gem Stapleton, Peter Rodgers, Pal diagrams: A linear diagram-based visual language, J. Vis. Lang. Comput. 25 (6) (2014) 945–954.
[13] S.-J. Shin, The Logical Status of Diagrams, Cambridge University Press, 1994.
[14] P. Chapman, G. Stapleton, P. Rodgers, L. Michallef, A. Blake, Visualizing sets: An empirical comparison of diagram types, in: Diagrams 2014, Springer, 2014, pp. 146–160.
[15] Gem Stapleton, Peter Chapman, Peter Rodgers, Anestis Touloumis, Andrew Blake, Aidan Delaney, The efficacy of euler diagrams and linear diagrams for visualizing set cardinality using proportions and numbers, PLoS One 14 (3) (2019) e0211234.
[16] Saturnino Luz, Masood Masoodian, A comparison of linear and mosaic diagrams for set visualization, Inf. Vis. 18 (3) (2019) 297–310.

[17] Peter Rodgers, Gem Stapleton, Peter Chapman, Visualizing sets with linear diagrams, ACM Trans. Comput.-Hum. Interact. 22 (6) (2015) 27.
[18] Gem Stapleton, Peter Rodgers, Anestis Touloumis, Andrew Blake, Well-matchedness in euler and linear diagrams, in: International Conference on Theory and Application of Diagrams, Springer, 2020, pp. 247–263.
[19] Peter Chapman, Interactivity in linear diagrams, in: Diagrammatic Representation and Inference - 12th International Conference, Diagrams 2021, Virtual, September 28-30, 2021, Proceedings, in: Lecture Notes in Computer Science, vol. 12909, Springer, 2021, pp. 449–465.
[20] Mohanad Alqadah, Gem Stapleton, John Howse, Peter Chapman, The perception of clutter in linear diagrams, in: Diagrammatic Representation and Inference, in: LNAI, vol. 9781, Springer, 2016, pp. 250–257.
[21] Mohanad Alqadah, Gem Stapleton, John Howse, Peter Chapman, Evaluating the impact of clutter in Euler diagrams, in: Diagrammatic Representation and Inference, in: LNAI, vol. 8578, Springer, 2014, pp. 108–122.
[22] Mohanad Alqadah, Gem Stapleton, John Howse, Peter Chapman, Evaluating the impact of clutter in linear diagrams, in: SetVR@ Diagrams, 2016, pp. 4–18.
[23] Lawrence T. Kou, Polynomial complete consecutive information retrieval problems, SIAM J. Comput. 6 (1) (1977) 67–75.
[24] Salim Haddadi, A note on the NP–hardness of the consecutive block minimization problem, Int. Trans. Oper. Res. 9 (6) (2002) 775–777.
[25] Leonardo C.R. Soares, Jordi Alves Reinsma, Luis H.L. Nascimento, Marco A.M. Carvalho, Heuristic methods to consecutive block minimization, Comput. Oper. Res. 120 (2020) 104948.
[26] Salim Haddadi, S. Chenche, Meryem Cheraitia, F. Guessoum, Polynomial-time local-improvement algorithm for consecutive block minimization, Inform. Process. Lett. 115 (6–8) (2015) 612–617.
[27] Konstantin Chakhlevitch, Celia A. Glass, Natalia V. Shakhlevich, Minimising the number of gap-zeros in binary matrices, European J. Oper. Res. 229 (1) (2013) 48–58.