

Constrained Hierarchical Clustering via Graph Coarsening and Optimal Cuts

Eliabelle Mauduit^{1,2}, Andrea Simonetto¹

¹ Unité de Mathématiques Appliquées, ENSTA Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France

² Greenworking, 110 rue de Picpus, 75012 Paris, France

Abstract—Motivated by extracting and summarizing relevant information in short sentence settings, such as satisfaction questionnaires, hotel reviews, and X/Twitter, we study the problem of clustering words in a hierarchical fashion. In particular, we focus on the problem of clustering with horizontal and vertical structural constraints. Horizontal constraints are typically cannot-link and must-link among words, while vertical constraints are precedence constraints among cluster levels.

We overcome state-of-the-art bottlenecks by formulating the problem in two steps: first, as a soft-constrained regularized least-squares which guides the result of a sequential graph coarsening algorithm towards the horizontal feasible set. Then, flat clusters are extracted from the resulting hierarchical tree by computing optimal cut heights based on the available constraints.

We show that the resulting approach compares very well with respect to existing algorithms and is computationally light.

Index Terms—Constrained Hierarchical Clustering, Graph Coarsening, Optimization, Ultrametric Spaces

I. INTRODUCTION

In modern society, extracting, classifying, and summarizing information from text is becoming of paramount importance. This is fuelled by the large amount of content that is available on social media platforms, and on the Internet in general. In this paper, we are particularly interested in extracting topics and clustering words in short sentences. This text modality is typical in satisfaction questionnaires, X/Twitter, and most of the online posts. Hierarchical clustering is widely used for genomics data [1], social network analysis [2], bioinformatics [3], text classification [4], and financial markets [5].

Specifically, we focus here on clustering words from short sentences in a hierarchical fashion and adding structural constraints. For most real-life applications, users might have prior information which is not contained in the input features, hence the interest to specify structural constraints. Since the constraints come from contextual information about the data, our problem falls in the category of semi-supervised clustering. Constrained versions of clustering and hierarchical clustering have appeared in the past [6]–[15] and the area stays very active [16], [17]. Here, we propose an efficient algorithm to combine both *horizontal constraints*, meaning must-link and cannot-link constraints between words, and *vertical constraints*, meaning precedence constraints among different layers, or levels, of the hierarchy. The aforementioned features make our algorithm novel and unique in its genre.

We overcome some of the bottlenecks encountered in the literature by considering a two-steps approach. **First**, we

develop a soft-constrained version of both vertical and horizontal constraints added as a regularization term to a graph coarsening algorithm. This enables to take the prior knowledge into account while respecting the initial structure of the data space, and makes it possible to deal with conflicting sets of constraints [13]. The result of this step is a dendrogram, which is a tree that iteratively splits a data set into smaller subsets until each subset consists of only one element. **Second**, the obtained dendrogram is then cut at different levels in order to get hierarchical flat clusters that meet as many constraints as possible. The key of our two-steps approach is to automatically get the desired flat clusters, as we will discuss shortly.

Another particular feature of our approach is that we will consider vertical and horizontal constraints per layer. Since in hierarchical clustering, all the data points will eventually belong to the same set, the constraints have to change or evolve layer per layer. Two data points may have a cannot link at a lower layer, but then they can be merged at a higher level. We call this algorithmic feature: layer-based constraints.

Related work. Hierarchical and semi-supervised clustering has been studied extensively in the past [17]. Most existing hierarchical clustering are based on linkage distance methods, and the main differential is in the way prior information is handled. The standard way is to add pairwise constraints [7], [18], [19] which are basically horizontal must-links and/or cannot-links. While such constraints are a good starting point, they hierarchical layer-based feature is missing. One way to add this is to consider triple-wise constraints instead as in [20]. However, this approach has the drawback that it does not handle cannot-links and does not guarantee that the merging heights of the different triplets will define clear levels.

On the constraint side, when hard constraints are imposed [18], lower complexities can be reached ($\mathcal{O}(n \log(n))$ where n is the number of data points) to the potential detriment of the geometry of the initial data space. However, hard constraints might lead to errors when the set of constraints is not well defined or infeasible. Thus, even though using soft constraints increases algorithmic complexity ($\mathcal{O}(n^2)$ for [19], [20] and $\mathcal{O}(n^3)$ for the alternative in [20]), they might be preferable to work with to avoid infeasibility.

Another traditional method to perform clustering is to use graph-based reduction algorithms. In such a framework, the space is represented as a graph, and among the different strategies we can cite single-graph [21], [22] and multiple graphs clustering [23]. Contrary to single graph approach

which works on one graph only and its Laplacian, multiple graphs techniques cluster the data points based on the knowledge provided by several graphs. This can enhance the performances of the clustering process compared to single-graph but it also increases the memory cost.

All the methods introduced above are either non hierarchical or do not result in flat clusters. To our knowledge, very few methods exist to perform automatic cluster extraction from a hierarchical structure. In their article [24], Jörg and co-authors introduce a method to deduce a hierarchical partitioning of the data based on the reachability plot resulting from the hierarchical clustering. Their algorithm is based on the identification of clear “dents” or “valleys” in the reachability plot and perform very well on examples where clusters are clearly separated in the initial data structure. Nevertheless, in our case, the data structure does not consist of clean and well defined (sub-)clusters and the algorithm from [24] cannot be directly applied.

Contributions. In this paper, we introduce a new semi-supervised algorithm to perform hierarchical flat clustering. Our algorithms emerge as the natural extensions of A. Loukas’s local-variation multi-level graph coarsening [25] and of the regular bottom-up clustering [26]. These methods rely on the Laplacian and on the distance matrix of the graph associated to the dataset to build the coarsening. We carefully adjust these matrices in order to favour clusters that meet as many constraints as possible. We show that the soft-constrained problem can be rewritten as a quadratic problem and that, in the case of [26], it can be solved in closed-form.

Our second contribution lies in the automation of the dendrogram (or ultrametric) interpretation. To do so, we propose a method that computes the optimal cuts of the dendrogram given the input structural constraints. The idea is to find, for each hierarchical level, the cut that minimizes the distance to the equivalent cut in the constraints tree. We prove that the resulting problem is convex and can be solved in closed-form.

We run experiments on the Hotel dataset [27] to demonstrate our methods and compare them to their unconstrained counterparts. The dataset consists of a collection of hotel reviews, and our goal is to run hierarchical clustering algorithms on the vocabulary of the reviews in order to extract topics and lexical fields in a hierarchical fashion. Extra information is provided in the form of hierarchical must-links and cannot-links. In that case, the addition of constraints led to up to a 67% improvement in terms of constraints compliance, and up to a 19% improvement in terms of Dasgupta’s cost [28] (a generally adopted metric in hierarchical clustering). These results support our algorithms.

II. STEP I : SEQUENTIAL GRAPH COARSENING

A. Problem description

Consider a set \mathcal{S} of n data points. The objective is to extract clusters of points sharing similar features from \mathcal{S} in a hierarchical fashion. In addition, layer-based constraints are provided by the user to add prior information on the expected clusters. In particular, the user provides a wished number of layers ℓ , and a constraint set \mathcal{C}_j , $j = 1, \dots, \ell$,

for each of them. Constraints are pairwise must and cannot-links between points that can be either horizontal (when the link relates to one or two clusters from the same hierarchical level) or vertical (to allow links between clusters of different granularity levels). Horizontal constraints are specified in \mathcal{C}_j , while vertical constraints are precedence constraints and can be specified as cannot-link in \mathcal{C}_j and subsequently as a must-link in \mathcal{C}_{j+1} .

We formalized the problem of hierarchical clustering with layer-based constraints by a two-steps approach, and this section is dedicated to the first stage, which is the construction of a dendrogram based on the data space \mathcal{S} and user’s prior knowledge.

Consider a graph representation of the dataset \mathcal{S} as $\mathcal{G} = (V, E, \mathcal{W})$, where each node in the vertex set V is data point and the edges’ weights $\mathcal{W}(E)$ correspond to the distance (or similarity) between the points. In this article, we will only consider the distance matrix \mathbf{D} and its associated Laplacian \mathbf{L} , but all the results hold in the similarity framework. Our hierarchical clustering problem consists in sequentially merging graph nodes in a *suitable* order until there is only one left. We adapted two existing methods by adding soft structural constraints. The first one, that we will call local-variation coarsening [25] is based on the graph Laplacian \mathbf{L} and chooses the reduction that minimizes spectral variations of \mathbf{L} . The second one is the traditional bottom-up hierarchical clustering [26] that sequentially merges the closest nodes and computes the new distance matrix based on a predefined linkage method. The main difference between both algorithms lies on the methodology used to pick nodes to merge and compute the new distance matrix. While bottom-up linkage merely groups the two closest nodes at each step, the local-variation coarsening simultaneously merges as many pairs as possible using an edge or neighborhood-based method.

Local-variation coarsening. We focus here on edge-based methods. Given an initial Laplacian \mathbf{L} consisting of N vertices, the result is a new Laplacian consisting of $\lfloor N/2 \rfloor$ vertices. Contraction is based on minimizing the spectral variation of the Laplacian before and after the contraction. The complexity of this algorithm is $\mathcal{O}(N \log N)$. The fact that the graph size is being halved at each iteration makes the method computationally lightweight. However, this might be too coarse-grained for specific datasets.

Bottom-up approach. This approach relies on linkage methods and the computation of the distance matrices. There are several ways to compute the new distance matrix after one contraction. Distance can be computed using one of the seven linkage methods (e.g., single, average). Naive implementations of these methods can have high complexities ($\mathcal{O}(N^3)$). However, using nearest-neighbour chains [29], a computational complexity of ($\mathcal{O}(N^2)$) can be reached.

B. Soft-constrained coarsening

In both methods, nodes are merged by pairs using either the local variation method or the linkage clustering. These contractions are fully determined by the Laplacian or the distance matrix hence the necessity of updating them by taking

constraints into account. In particular, given the coarsened Laplacian operator $\mathbf{L}_c \in \mathbb{R}^{m \times m}$, where $m < N$, coming from one coarsening step of one of the two methods, our algorithm modifies it to satisfy as many constraints as possible. The latter is achieved by solving the problem,

$$\begin{aligned} P(\mathbf{L}_c) : \min_{\mathbf{L} \in \mathcal{L} \subset \mathbb{R}^{m \times m}} & \frac{1}{2} \|\mathbf{L} - \mathbf{L}_c\|^2 + \\ & + \frac{\lambda_1}{2} \sum_{(i,j) \in \text{ML}} \|\mathbf{L}_{i,j}\|^2 + \frac{\lambda_2}{2} \sum_{(i,j) \in \text{CL}} \|\mathbf{L}_{i,j} + 1\|^2, \end{aligned} \quad (1)$$

where the set ML represents the must links, and CL the cannot links, λ_1, λ_2 are two positive scalar weights, \mathcal{L} is the set of Laplacian matrices (symmetric, with rows summing to zero and negative elements except on the diagonal), and $\|\cdot\|$ is the Frobenius norm.

Each layer-based constraint set \mathcal{C}_j contains constraints in the form of constraints matrices ML_j and CL_j whose entries are 1 if there is a must-link or cannot-link, respectively, between datapoints, and zero otherwise.

Problem $P(\mathbf{L}_c)$ is a convex quadratic program, which can be solved efficiently by off-the-shelf solvers. An interesting feature of the bottom-up approach is that the constraint $\mathbf{L} \in \mathcal{L}$ is less important in practice and can be substituted with $\mathbf{L} \in [-1, 0]^{m \times m}$. In fact, one can focus on the upper triangle of the Laplacian \mathbf{L}_c alone, and reconstruct the Laplacian a posteriori, if needed. This practical simplification, sensible only for the bottom-up approach, renders $P(\mathbf{L}_c)$ solvable in closed-form, since each entry is now independent, as formalized next.

Proposition 1 (Practical simplification for the bottom-up approach). *Consider Problem $P(\mathbf{L}_c)$, replacing the constraint $\mathbf{L} \in \mathcal{L}$ with $\mathbf{L} \in [-1, 0]^{m \times m}$. Define $\lambda_{1,ij} = 1$ if $(i, j) \in \text{ML}$, and 0 otherwise, and do the same for $\lambda_{2,ij} = 1$ for the cannot links. This new problem can be solved in closed-form as,*

$$\mathbf{L}_{ij}^* = \min \left\{ \max \left\{ \frac{\mathbf{L}_{c,ij} - \lambda_{2,ij}}{1 + \lambda_{1,ij} + \lambda_{2,ij}}, -1 \right\}, 0 \right\}, \quad \forall j > i.$$

Proof. By optimality conditions. \square

The first step of our algorithm consists in iteratively solving $P(\mathbf{L}_c)$ or its simplified version, and doing a coarsening pass, multiple times till most of the constraints in \mathcal{C}_j for layer j are satisfied and we cannot reduce the graph anymore. Then, we continue with a new layer set \mathcal{C}_{j+1} , and so on.

The final output is a large dendrogram where points connected by must-links are likely merged at a low heights and those connected by cannot-links are merged at a later layer. From there, our aim is to deduce hierarchical flat clusters without parsing the whole tree by hand. In the next section, we introduce a systematic approach to obtain such clusters given a dendrogram and layer-based constraints.

III. STEP II: OPTIMAL HIERARCHICAL CUTS

A. Problem Description

Given a dendrogram, the problem we want to solve is how to obtain hierarchical flat clusters satisfying the horizontal and vertical constraints. The usual approach would be to parse the

dendrogram manually, but this is hardly an option for large-scale trees. Another possibility is to cluster from the layers that we have built. But this may be sub-optimal, since not all the constraints may have been satisfied. Here we propose a different approach.

Consider ℓ layers. For each layer $j \in 1, 2, \dots, \ell$, we consider the clusters that Step I has produced. These are indicated with $c_{i,j}, i = 1, \dots, n_i$ as for cluster i , in layer j . As we are parsing a dendrogram, which is a binary tree, we can order the branches from the data at height 0 to the root at height $H > 0$. We can then define two heights for each $c_{i,j}$.

The first is $h_{i,j}$ as the minimal height at which all the must-link of $c_{i,j}$ are satisfied. If Step I is completely successful, the height $h_{i,j}$ will correspond to the height derived from Step I, otherwise it will be higher. By construction, we know that such height exists, since eventually all the data will be merged in a single point at H . The second height is $H_{i,j}$, defined as the minimal height greater or equal than $h_{i,j}$ at which one cannot-link of $c_{i,j}$ is violated. Once again, if Step I is successful $H_{i,j}$ could be equivalent to $h_{i,j}$ or strictly higher. If $c_{i,j}$ has no cannot link, then $H_{i,j} = H$.

Then, we define the optimal level cut as the problem of finding the height at which to cut the dendrogram for each level j , such that we minimize constraint violation,

$$\tilde{h}_j \in \operatorname{argmin}_{h \in \mathbb{R}_+} \sum_{i=1}^{n_i} \mathcal{D}_{i,j}(h), \quad (2)$$

where $\mathcal{D}_{i,j} : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is the function defined as the distance between h and its convex projection onto the interval $[h_{i,j}, H_{i,j}]$, that is $\mathcal{D}_{i,j}(h) = |h - \max\{\min\{h, H_{i,j}\}, h_{i,j}\}|$.

Proposition 2. *Let $\hat{h}_1 \leq \hat{h}_2 \leq \dots \leq \hat{h}_{2n_i}$ be the elements of $\{h_{1,j}, H_{1,j}, \dots, h_{n_i,j}, H_{n_i,j}\}$ sorted in increasing order. Problem 2 is convex and any point of the set $\{h | h \in [\hat{h}_{n_i}, \hat{h}_{n_i+1}]\}$ is a solution for (2).*

Proof. (Sketch) Convexity of (2) follows from the convexity of $\mathcal{D}_{i,j}$, which can be proved by construction. Then, writing the optimality conditions, the solution set can be derived. \square

As in Step I, Step II is based on a problem that can be solved in closed-form and its resolution is computationally lightweight. We are now ready to discuss the overall algorithm.

B. Overall algorithm

We now present our main algorithm, labeled Clues, as to indicate that we are *clustering* with soften prior information (that is: clues), we describe it in Algorithm 1, and it performs hierarchical flat clusters extraction using the soft-constrained Problem 1 and the optimal cuts given by Prop. 2. Clues is also depicted in Figure 1.

If the method is the traditional bottom-up clustering, the algorithm works by locally looking at the distance matrix values and treating edges independently. Hence, using the closed-form alternative given in Prop. 1 leads to satisfying results and helps keeping control of the complexity (which is $\mathcal{O}(n^3)$, and $\mathcal{O}(n^2)$ when using nearest-neighbour chains).

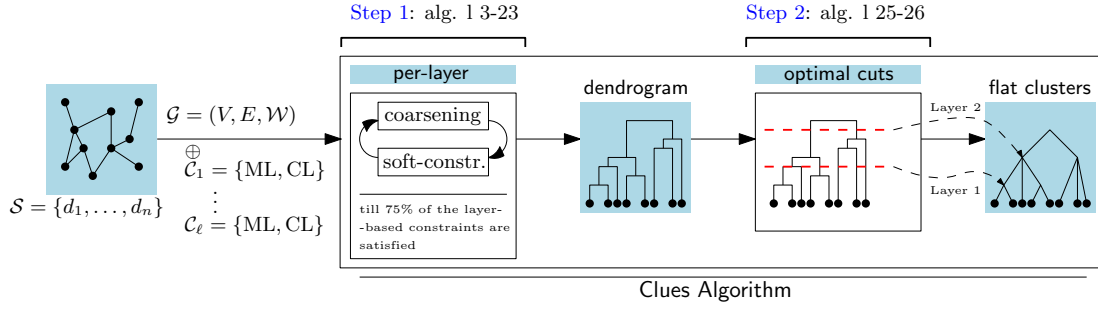


Fig. 1: Schematic diagram of Clues' main steps.

Algorithm 1 Clues

Input: data set \mathcal{S} of n data points, distance matrix \mathbf{D} , smoothing parameters λ_1, λ_2 , level-based pairwise constraints \mathcal{C}_j , coarsening method, $I_{\max} > 0$

Output: Hierarchical flat clusters

- 1: Build the graph \mathcal{G}
- 2: // Step I
- 3: **for** Layers $k = 1, \dots, j$ **do**
- 4: $I = 0$
- 5: **if** method is traditional bottom up clustering **then**
- 6: Update \mathbf{D} using Prop. 1
- 7: **while** 75% of \mathcal{C}_j is not satisfied and $I < I_{\max}$ **do**
- 8: Merge the two closest nodes
- 9: Compute the new distance matrix \mathbf{D}
- 10: Update the new \mathbf{D} using Prop. 1
- 11: Update \mathcal{G} , $I \leftarrow I + 1$
- 12: **end while**
- 13: **else** ▷ method is local variation coarsening
- 14: Build the Laplacian \mathbf{L}
- 15: Update \mathbf{L} using Problem 1
- 16: **while** 75% of \mathcal{C}_j is not satisfied and $I < I_{\max}$ **do**
- 17: Merge as many pairs of nodes as possible
- 18: Compute the new Laplacian \mathbf{L}
- 19: Update \mathbf{L} using Problem 1
- 20: Update \mathcal{G} , $I \leftarrow I + 1$
- 21: **end while**
- 22: **end if**
- 23: **end for**
- 24: // Step II
- 25: Compute the set of optimal cuts \mathcal{C} of the dendrogram \mathcal{D} resulting from the clustering using Prop. 2
- 26: Cut \mathcal{D} using \mathcal{C} and derive the hierarchical flat clusters

As for the local variation clustering, we solve Problem 1 at each step to exploit the neighborhood information contained in the diagonal term. The worst-case complexity is $\mathcal{O}(n^3)$.

IV. RESULTS

A first example. We test our algorithm on an hotel reviews database [27], with the aim of comparing themes and lexical fields across the different states in the USA. The data consist of n reviews and meta-data containing, e.g., timestamps and locations. We use the former to build our vocabulary and dendrogram and the latter to locate the hotel.

Example trees of hierarchical topic extraction for New York and Arizona reviews are displayed in Figure 2, setting two layers. Clusters labels were built manually from the output lexical fields. At this granularity level, clusters are similar for

both states as they are guided by the constraints. Nonetheless, we can already notice a novel theme (inclusiveness) and sub-theme (temperature) in the Arizona tree that were not defined in the constraints (indicated in blue on the trees). Thus, using constraints to give a hierarchical structure that guides the algorithm helps achieving consistent results while still allowing the extraction of novel information.

The main source of information lies in the vocabulary associated with each cluster. Labels might be subjective or poorly set, hence the necessity to check results at the finest level. For instance, even though labels are the same, the lexical fields differ according to the state. In Figure 2, we have also displayed the lexical fields behind “neighbourhood attractiveness”. While both related to the interests of the hotel area, the lexical fields are different and illustrate the heterogeneity in tourists’ expectations according to the area. Note that some words in the lexical fields (in red) are inconsistent or meaningless. These words constitute errors of the algorithm.

Comparisons. We move to assess the performances of Clues based on three different criteria: (i) the widely used Dasgupta’s cost [28] that measures the resulting dendrogram “quality” (the lower the cost the better); (ii) the percentage of violated constraints; and (iii) the runtime of the algorithm. We also compare Clues to its unconstrained version, where we do not perform the soft-constrained coarsening in Step I, i.e., we use the algorithms of [25] and [26] as they are.

As depicted in Figure 3, where we use 50 different review datasets, Clues obtains the best results for constraints enforcement (leading to up to a 67% improvement, and a below 5% overall constraint violation), and it can further reduce the Dasgupta’s cost in the bottom-up approach (up to a 19% improvement). Time-wise, we use the size of the dots to represent the normalized run-time, and we can see that the local-variation version of Clues is the most demanding. This is due to the use of CVX to solve the soft-constrained problem (1), and it represents a $\sim 3\times$ increase in computational time.

The 50 datasets are extracted from the hotel reviews sorted by state. This enables to have sets of different sizes (from ~ 25 to ~ 1250 points) and varied vocabulary while keeping the same constraints.

V. CONCLUSIONS

In this paper, we introduced an approach to perform semi-supervised hierarchical clustering on large databases. The

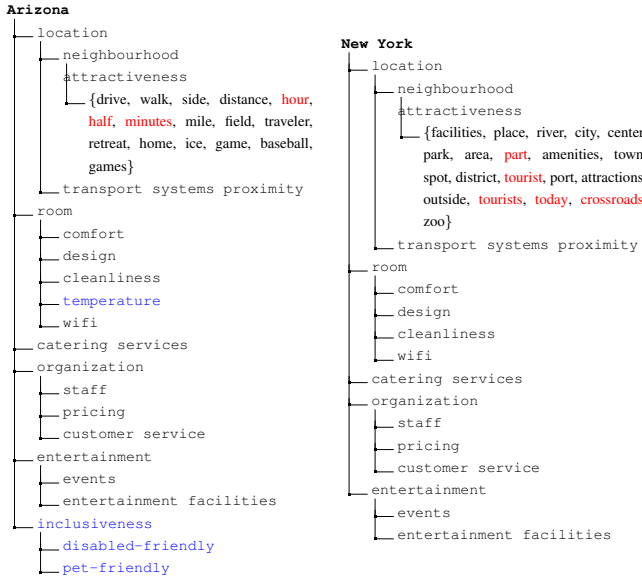


Fig. 2: Hierarchical topics extraction running Clues on hotel reviews. Topics names have been inferred manually from the lexical fields. An example is given for the sub-cluster “attractiveness” for comparative purposes.

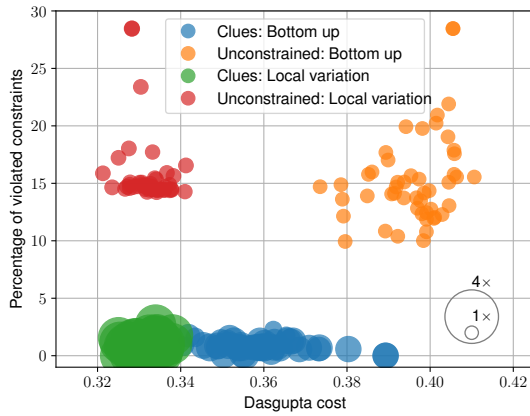


Fig. 3: Performance of Clues and comparison: lower Dasgupta costs are associated with higher quality hierarchical clustering hence points on the lower left represent the best results. The sizes of the points are proportional to the normalized time taken to run Clues on the corresponding dataset.

specificity of our method being the possibility to extract flat clusters automatically from word data by exploiting structural and user-defined constraints, making it easier to analyze the results. Constraints have to be provided in the form of layer-based must-links and cannot-links and are used throughout the whole process to guide the clustering and extract the final flat non-binary tree. Performances of our method (named Clues) are evaluated on a benchmark dataset and the results encourage further work in this direction.

REFERENCES

- [1] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, “Cluster analysis and display of genome-wide expression patterns,” *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, 1998.
- [2] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive data sets*. Cambridge university press, 2020.
- [3] I. Diez, P. Bonifazi, I. Escudero, B. Mateos, M. A. Muñoz, S. Stramaglia, and J. M. Cortes, “A novel brain partition highlights the modular skeleton shared by structure and function,” *Scientific reports*, vol. 5, no. 1, p. 10532, 2015.
- [4] M. Steinbach, G. Karypis, and V. Kumar, “A comparison of document clustering techniques,” *KDD workshop on text mining*, vol. 400, pp. 525–526, 2000.
- [5] M. Tumminello, F. Lillo, and R. N. Mantegna, “Correlation, hierarchies, and networks in financial markets,” *Journal of Economic Behavior & Organization*, vol. 75, pp. 40–58, jul 2010.
- [6] S. Basu, A. Banerjee, and R. J. Mooney, “Active semi-supervision for pairwise constrained clustering,” in *Proceedings of the 2004 SIAM international conference on data mining*, pp. 333–344, SIAM, 2004.
- [7] I. Davidson and S. Ravi, “Agglomerative hierarchical clustering with constraints: Theoretical and empirical results,” in *Knowledge Discovery in Databases: PKDD 2005: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3-7, 2005. Proceedings 9*, pp. 59–70, Springer, 2005.
- [8] K. Bade and A. Nurnberger, “Personalized hierarchical clustering,” in *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI’06)*, pp. 181–187, IEEE, 2006.
- [9] J. Kawale and D. Boley, “Constrained spectral clustering using l1 regularization,” in *Proceedings of the 2013 SIAM International Conference on Data Mining*, pp. 103–111, SIAM, 2013.
- [10] E. Bair, “Semi-supervised clustering methods,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 5, no. 5, pp. 349–361, 2013.
- [11] R. Florence, B. Nogueira, and R. Marcacini, “Constrained hierarchical clustering for news events,” in *Proceedings of the 21st International Database Engineering & Applications Symposium*, pp. 49–56, 2017.
- [12] G. Carlsson, F. Mémoli, A. Ribeiro, and S. Segarra, “Hierarchical clustering of asymmetric networks,” *Advances in Data Analysis and Classification*, vol. 12, pp. 65–105, 2018.
- [13] V. Chatziafratis, R. Niazadeh, and M. Charikar, “Hierarchical clustering with structural constraints,” in *International conference on machine learning*, pp. 774–783, PMLR, 2018.
- [14] W. Huang and A. Ribeiro, “Hierarchical clustering given confidence intervals of metric distances,” *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2600–2615, 2018.
- [15] C.-L. Liu, W.-H. Hsiao, T.-H. Chang, and H.-H. Li, “Clustering data with partial background information,” *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 1123–1138, 2019.
- [16] D. Bakkelund, “Order preserving hierarchical agglomerative clustering,” *Machine Learning*, vol. 111, no. 5, pp. 1851–1901, 2022.
- [17] J. Cai, J. Hao, H. Yang, X. Zhao, and Y. Yang, “A review on semi-supervised clustering,” *Information Sciences*, 2023.
- [18] K. L. Wagstaff and C. Cardie, “Clustering with instance-level constraints,” in *AAAI/IAAI*, 2000.
- [19] T. Yang, N. Pasquier, and F. Precioso, “Semi-supervised consensus clustering based on closed patterns,” *Knowledge-Based Systems*, vol. 235, p. 107599, 2022.
- [20] L. Zheng and T. Li, “Semi-supervised hierarchical clustering,” in *IEEE 11th International Conference on Data Mining*, pp. 982–991, 2011.
- [21] W. Chen and G. Feng, “Spectral clustering: A semi-supervised approach,” *Neurocomputing*, vol. 77, no. 1, pp. 229–242, 2012.
- [22] T. Semertzidis, D. Rafailidis, M. Strintzis, and P. Daras, “Large-scale spectral clustering based on pairwise constraints,” *Information Processing & Management*, vol. 51, no. 5, pp. 616–624, 2015.
- [23] S. Liu, C. Ding, F. Jiang, Y. Wang, and B. Yin, “Auto-weighted multi-view learning for semi-supervised graph clustering,” *Neurocomputing*, vol. 362, pp. 19–32, 2019.
- [24] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky, “Automatic extraction of clusters from hierarchical clustering representations,” in *Advances in Knowledge Discovery and Data Mining*, pp. 75–87, Springer, 2003.
- [25] A. Loukas, “Graph reduction with spectral and cut guarantees,” *J. Mach. Learn. Res.*, vol. 20, no. 116, pp. 1–42, 2019.
- [26] D. Müllner, “Modern hierarchical, agglomerative clustering algorithms,” *arXiv preprint arXiv:1109.2378*, 2011.
- [27] A. Modi, “Hotel reviews data science.” https://github.com/abhikasd6523/HotelReview_DataScience/blob/master/Hotel_Reviews.csv, 2017.
- [28] S. Dasgupta, “A cost function for similarity-based hierarchical clustering,” in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 118–127, 2016.
- [29] F. Murtagh, *Multidimensional clustering algorithms*. Physica-Verlag, 1985.