Agglomerative Info-Clustering: Maximizing Normalized Total Correlation

Chung Chan, Ali Al-Bashabsheh and Qiaoqiao Zhou

Abstract

We show that, under the info-clustering framework, correlated random variables can be clustered in an agglomerative manner in addition to a divisive one. While the original divisive approach successively segregates the random variables into subsets with increasing multivariate mutual information, our agglomerative approach successively merges subsets of random variables sharing a large amount of normalized total correlation. We implemented the algorithm and showed that it gives the same hierarchy of clusters faster than the divisive approach by an order of magnitude. The equivalence of the two approaches is a fundamental and meaningful connection between the well-known total correlation and the recently proposed measure of multivariate mutual information.

Index Terms

multivariate mutual information; agglomerative clustering; principal sequence of partitions; principal sequence; minimum norm base

I. Introduction

In hierarchical clustering analysis, there are two methods of generating a hierarchy of clusters [2, 3]: divisive and agglomerative. The divisive approach starts with the trivial cluster consisting of all the elements and iteratively breaks a large cluster into smaller ones. The agglomerative approach starts with the singleton clusters instead and group smaller clusters into larger ones. Common to both approaches is the use of a proximity measure that drives the clustering process. For agglomerative clustering, each merging step attempts to minimize the separation (maximize the similarity) of the clusters to be merged. For divisive clustering, each partitioning step attempts to maximize the separation (minimize the similarity) of the resulting clusters. Different measures of proximity may be used, resulting in different algorithms [4–6]. Intuitively, if the same measure of proximity is used, divisive and agglomerative methods should return the same clustering solution. Such consistency or duality between divisive and

Preliminary work of this paper was presented at [1].

C. Chan (corresponding author, email: chung.chan@cityu.edu.hk) is with the Department of Computer Science, City University of Hong Kong. His work was supported by a grant from the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. 21203318).

A. Al-Bashabsheh is with the Big Data and Brain Computing (BDBC) center at Beihang University, Beijing, China (e-mail: entropy-ali@gmail.com).

Q. Zhou is with the Institute of Network Coding and the Department of Information Engineering, the Chinese University of Hong Kong.

agglomerative clustering is essential for a rigorous hierarchical clustering formulation. However, to the best of our knowledge, such duality is not required for the choice of proximity measures. In other words, many agglomerative clustering algorithms may not have their divisive counterparts and vice versa. An example is the complete-linkage method, where, setting aside the lack of an efficient complete-linkage divisive algorithm, the agglomerative and divisive approaches may result in different hierarchies. Complete-linkage measures the separation between two clusters by the furthest distance of points in the two clusters. If the closest distance is used instead of the furthest distance, we have the single-linkage method, which indeed gives a unique hierarchy and satisfies the desired duality. However, compared to the complete-linkage method, it is known that the single-linkage method cannot capture higher-order linkage between objects. A question of interest is whether there is an extension of the single-linkage method that has consistent divisive and agglomerative counterparts.

In this work, we consider the info-clustering framework as a meaningful extension of the single-linkage method that can capture higher-order linkage while maintaining consistency of the divisive and agglomerative approaches. Info-clustering was proposed in [7] as a hierarchical clustering of random variables (RVs) based on their multivariate mutual information (MMI) [8]. The hierarchy of clusters is unique and was characterized in [7] as the principal sequence of partitions (PSP) [9] of the entropy function of the RVs. This leads to a polynomial-time algorithm [7, Algorithm 3] that computes the clustering solution in $O(n^2\,\mathrm{SFM}(n))$ time, where n is the number of RVs and $\mathrm{SFM}(n)$ is the time required to minimize a submodular function on a ground set of size n. Beyond clustering, the algorithm can be motivated from a purely information-theoretic point of view since, as a byproduct, it also computes the MMI of the RVs. The current implementation of the MMI computation [10] requires an exponential time in n. An implementation of the polynomial-time info-clustering algorithm is therefore much desired.

While one can implement the polynomial-time info-clustering algorithm in [7, Algorithm 3], in practice: 1) One may want to obtain clusters of a desired size rather than the entire hierarchy of clusters of different sizes. 2) The entropy function needs to be estimated from data, which can be difficult for a large set of random variables. Although this work assumes an oracle for the entropy function, these practical considerations (or issues) motivate the following question: Can we construct an info-clustering algorithm that a) has the same complexity of $O(n^2\,\mathrm{SFM}(n))$ as [7, Algorithm 3], and b) computes the clusters iteratively according to their sizes? Note that due to the 2nd practical consideration, i.e., to reduce the chances of propagating errors arising from unreliable entropy estimations, it is preferable if the iterative algorithm starts with smaller clusters and proceeds to larger ones. A divisive info-clustering algorithm was given in [7, Algorithms 1 & 2] that breaks down the computation by subdividing the entire set of RVs successively into increasingly smaller clusters. However, this slows down the clustering-solution computation by an order of n (compared to [7, Algorithm 3]), and may also suffer significant error propagations due to the 2nd practical consideration.

Another contribution of this work is an affirmative answer to the above question. Namely, we provide an agglomerative counterpart, with the desired complexity, that computes the clusters by grouping the RVs into increasingly larger clusters. The idea is inspired by the duality between the PSP and a related structure called the principal sequence (PS) [11–13]. We clarify these mathematical structures, which leads to a fundamental property

connecting the MMI and the well-known Watanabe's total correlation.

We remark that this work focuses on info-clustering. The clustering problem is a prominent one in data mining with vast literature and several clustering techniques. For a brief survey on the subject (that compares info-clustering to other clustering methods) we refer the interested reader to [7] and the references therein. Finally, similar to existing algorithms for computing the MMI or the info-clustering solution, this work assumes the entropy function is readily available. Nevertheless, this work can be viewed as a step towards a more practical implementation since it tries to mitigate the effects of the curse of dimensionality, faced by several entropy computation / estimation algorithms, by operating in an agglomerative manner. The problem of entropy estimation, or more generally functionals of the probability distribution, is well-studied and lies beyond the scope of this work. We refer the interested reader to, e.g., [14–17] and some of the references therein.

The paper is organized as follows. We first introduce the info-clustering formulation in Section II and the divisive approach in Section III. Section IV contains the preliminaries of the submodular optimization techniques for the main results in Section V. To demonstrate the feasibility of the agglomerative info-clustering algorithms, we provides an implementation in C++ on GitHub

https://github.com/ccha23/Agglomerative-Info-Clustering

and a Jupyter notebook that executes the code on Binder [18]. Section VI explains the implementation details, followed by the conclusion in Section VII.

II. Problem formulation

Consider a random vector $\mathsf{Z}_V := (\mathsf{Z}_i \mid i \in V)$ where $V := \{1, \dots, n\}$ is a set of $n > 1$ RVs. Let $\Pi(V)$ be the set of partitions of V into non-empty disjoint sets and $\Pi'(V) := \Pi(V) \setminus \{V\}$ be the set of non-trivial partitions. For $\mathcal{P}, \mathcal{P}' \in \Pi(V)$, we say $\mathcal{P}$ is finer than $\mathcal{P}'$, denoted $\mathcal{P} \preceq \mathcal{P}'$, if

$$\forall C \in \mathcal{P}, \exists C' \in \mathcal{P}' : C \subseteq C', \tag{2.1}$$

and use "$\prec$" to denote the strict inequality, i.e., when the inclusion above is strict for at least one $C \in \mathcal{P}$. The *multivariate mutual information* (MMI) [8] can be defined as

$$I(\mathsf{Z}_V) := \min_{\mathcal{P} \in \Pi'(V)} I_{\mathcal{P}}(\mathsf{Z}_V), \quad \text{where} \tag{2.2a}$$

$$I_{\mathcal{P}}(\mathsf{Z}_V) := \frac{1}{|\mathcal{P}| - 1} \left[\sum\nolimits_{C \in \mathcal{P}} H(\mathsf{Z}_C) - H(\mathsf{Z}_V) \right]. \tag{2.2b}$$

It follows that the MMI $I(\mathsf{Z}_V)$ is non-negative, and is equal to zero iff $P_{\mathsf{Z}_V} = \prod_{C \in \mathcal{P}} P_{\mathsf{Z}_C}$ for some $\mathcal{P} \in \Pi'(V)$.

Subsequently, we will denote the entropy function by h, i.e.,

$$h(B) := H(\mathsf{Z}_B) \qquad \text{for } B \subseteq V. \tag{2.3}$$

Essential to this work is the submodularity of entropy [19] or, equivalently, the non-negativity of the conditional mutual information [20]. More precisely, the *submodularity* of entropy means that the entropy function h in (2.3) satisfies

$$h(B_1) + h(B_2) \geq h(B_1 \cup B_2) + h(B_1 \cap B_2) \tag{2.4}$$

for all $B_1, B_2 \subseteq V$. The entropy function is also said to be *normalized* since $h(\emptyset) = 0$, and non-decreasing since $h(B') \leq h(B)$ whenever $B' \subseteq B \subseteq V$.

Using the MMI, the set of clusters at any threshold $\gamma \in \mathbb{R}$ is defined in [7] as

$$\mathcal{C}_\gamma(\mathsf{Z}_V) := \text{maximal}\{B \subseteq V \mid |B| > 1, I(\mathsf{Z}_B) > \gamma\}, \tag{2.5}$$

where $\text{maximal}\, \mathcal{F} := \{B \in \mathcal{F} \mid \nexists B' \in \mathcal{F}, B \subsetneq B'\}$ denotes the collection of inclusion-wise maximal elements of any collection $\mathcal{F}$ of subsets. It was shown in [7, Theorem 3] that the clusters form a laminar family, i.e., for any $\gamma' \leq \gamma''$, $C' \in \mathcal{C}_{\gamma'}(\mathsf{Z}_V)$, and $C'' \in \mathcal{C}_{\gamma''}(\mathsf{Z}_V)$, we have $C' \cap C'' = \emptyset$ or $C' \supseteq C''$. In particular, clusters at the same threshold must be disjoint. Consequently, the clustering solution can be characterized by a sequence of partitions of V as follows.

Proposition 2.1 ([7, Theorems 1 & 4]) *The clustering solution* (2.5) *satisfies*

$$\mathcal{C}_\gamma(\mathsf{Z}_V) = \mathcal{P}_\ell \setminus \{\{i\} \mid i \in V\} \;\; \forall \gamma \in [\gamma_\ell, \gamma_{\ell+1}), 0 \leq \ell \leq N, \tag{2.6a}$$

where N is a positive integer,

$$\gamma_0 := -\infty < \gamma_1 < \cdots < \gamma_N < \gamma_{N+1} := \infty \tag{2.6b}$$

is a sequence of distinct critical values from $\mathbb{R}$ (consisting of the thresholds at which the set of clusters changes), and

$$\mathcal{P}_0 := \{V\} \succ \mathcal{P}_1 \succ \cdots \succ \mathcal{P}_{N-1} \succ \mathcal{P}_N := \{\{i\} \mid i \in V\} \tag{2.6c}$$

is a sequence of increasingly finer partitions of V from $\Pi(V)$. □

Remark 1 The proposition above follows from the following property of the MMI

$$I(\mathsf{Z}_{B_1 \cup B_2}) \geq \min\{I(\mathsf{Z}_{B_1}), I(\mathsf{Z}_{B_2})\} \tag{2.7}$$

for all $B_1, B_2 \subseteq V : |B_1| > 1$, $|B_2| > 1$, and $B_1 \cap B_2 \neq \emptyset$. In other words, replacing $I(\mathsf{Z}_B)$ in (2.5) with any multivaraite information measure that satisfies (2.7), the resulting clusters will satisfy the proposition. However, for definiteness, we restrict attention to the MMI (2.2) in this work. In this case, the sequence of partitions in the proposition coincides with the principal sequence of partitions of the entropy function, as briefly discussed below.□

The sequence of partitions in Proposition 2.1 (together with the critical values) was further shown in [7, Corollary 2] to be the *principal sequence of partitions* (PSP) of the entropy function of Z_V, which arises from the Dilworth truncation of the submodular function h [9]. (For completeness, we give a brief discussion on the Dilowth truncation in Appendix A and refer the interested reader to [7–9] for more details.)

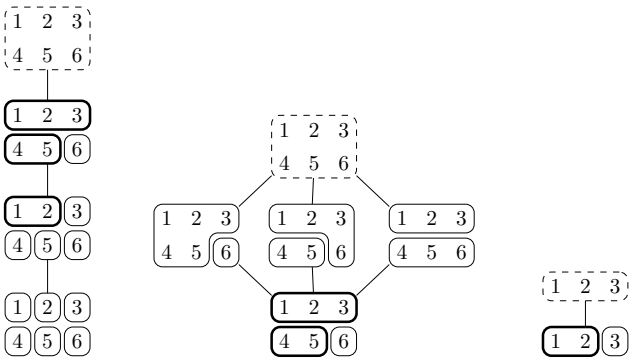


Fig. 1: (a) The PSP of Example 1, where the non-trivial clusters are circled with thick lines. (b) and (c) Optimal partitions of $I(\mathsf{Z}_V)$ and $I(\mathsf{Z}_{\{1,2,3\}})$, respectively, of Example 2. In (b) and (c), the fundamental partition is the one at the bottom and the associated clusters are circled with thick lines.

Example 1 As an illustration of Proposition 2.1 and the PSP, consider the following RVs defined, using the uniformly distributed and independent binary RVs $\mathsf{X}_a, \mathsf{X}_b, \mathsf{X}_c$ and X_d, as

$$\begin{aligned} &\mathsf{Z}_1 := (\mathsf{X}_a, \mathsf{X}_d), \quad \mathsf{Z}_2 := (\mathsf{X}_a, \mathsf{X}_d), \quad \mathsf{Z}_3 := \mathsf{X}_a, \\ &\mathsf{Z}_4 := \mathsf{X}_b, \qquad\quad \mathsf{Z}_5 := \mathsf{X}_b, \qquad\quad \mathsf{Z}_6 := \mathsf{X}_c. \end{aligned} \tag{2.8}$$

The PSP of the entropy function of $\mathsf{Z}_{\{1,\ldots,6\}}$ is shown in Fig. 1a, where the critical values are $\gamma_1 = 0, \gamma_2 = 1$, and $\gamma_3 = 2$. From Proposition 2.1, the clusters (2.5) are given as

$$\mathcal{C}_\gamma(\mathsf{Z}_{\{1,\ldots,6\}}) = \begin{cases} \{\{1,2,3,4,5,6\}\}, & \gamma < 0 \\ \{\{1,2,3\},\{4,5\}\}, & \gamma \in [0,1) \\ \{\{1,2\}\}, & \gamma \in [1,2) \\ \emptyset, & \gamma \geq 2. \end{cases} \tag{2.9}$$

□

From Proposition 2.1, given the PSP of the entropy function, one can readily obtain the clustering solution, and vice versa. There are algorithms for computing the PSP, see e.g., [7, Algorithm 3], [21, Ch. 13], [22]. However, such algorithms compute the partitions in the PSP in no particular order. Due to the practical considerations discussed earlier, it is desirable to have an algorithm that computes the PSP in some order. (From $\mathcal{P}_1$ to $\mathcal{P}_N$, or, preferably, from $\mathcal{P}_N$ to $\mathcal{P}_1$.)

III. Divisive approach

With the laminar structure in Proposition 2.1, a divisive algorithm was given in [7] to compute γ_ℓ and $\mathcal{P}_\ell$ by iteratively producing finer partitions from coarser ones, i.e., from $\mathcal{P}_1$ to $\mathcal{P}_N$. The following result is instrumental in the devising of such an algorithm.

Proposition 3.1 ([8, Theorems 5.2 & 5.3], [7, Theorem 4]) *The optimal partitions achieving the MMI in (2.2) together with the trivial partition $\{V\}$ form a lattice w.r.t. (2.1). The minimum/finest optimal partition, called the fundamental partition and denoted by $\mathcal{P}^*(\mathsf{Z}_V)$, satisfies*

$$\mathcal{P}^*(\mathsf{Z}_V)\backslash\{\{i\} \mid i \in V\} = \mathcal{C}_{I(\mathsf{Z}_V)}(\mathsf{Z}_V),$$

with $\mathcal{C}_\gamma$ defined in (2.5). In other words, $\gamma_1 = I(\mathsf{Z}_V)$ and $\mathcal{P}_1 = \mathcal{P}^*(\mathsf{Z}_V)$ in Proposition 2.1. Finally, for $\ell \geq 1$, if $\mathcal{P}_l$ is not the partition into singletons, then $\gamma_{\ell+1} = \min_{C\in\mathcal{P}_\ell:|C|>1} I(\mathsf{Z}_C)$ and $\mathcal{P}_{\ell+1} \supseteq \mathcal{P}^*(\mathsf{Z}_C)$ for any minimizer C. □

Because of the last statement in the proposition, any algorithm for computing $\mathcal{P}^*(\mathsf{Z}_V)$ can be applied iteratively for the divisive info-clustering approach as in [7, Algorithm 2]. However, as discussed near the end of Section I, this divisive approach can be a factor n slower than [7, Algorithm 3].

Example 2 As an illustration of Proposition 3.1 and the divisive approach, consider $\mathsf{Z}_{\{1,\ldots,6\}}$ in (2.8). The finest optimal partition $\mathcal{P}^*(\mathsf{Z}_V) = \{\{1,2,3\},\{4,5\},\{6\}\}$ is shown in Fig. 1b, and so one can easily verify from (2.2a) that $I(\mathsf{Z}_V) = 0$. For completion, the figure also shows the lattice of optimal partitions stated in the proposition, where the trivial partition $\{V\}$ is indicated using a dashed line. Similarly, Fig. 1c shows the optimal partitions of $\mathsf{Z}_{\{1,2,3\}}$. Since $I(\mathsf{Z}_V) = 0$, Proposition 3.1 asserts that $\mathcal{C}_0(\mathsf{Z}_V) = \{\{1,2,3\},\{4,5\}\}$. Since $I(\mathsf{Z}_{\{1,2,3\}}) = I(\mathsf{Z}_{\{4,5\}}) = 1$ (and $\{1,2,3\}$ is the only cluster with a non-singleton in its finest optimal partition), Proposition 3.1 asserts that $\mathcal{C}_1(\mathsf{Z}_V) = \mathcal{C}_1(\mathsf{Z}_{\{1,2,3\}}) = \{\{1,2\}\}$.

Assuming an algorithm for computing the finest optimal partition, the divisive algorithm starts by computing $\mathcal{P}^*(\mathsf{Z}_V) = \{\{1,2,3\},\{4,5\},\{6\}\}$ (from which $I(\mathsf{Z}_V)$ is readily available), declares the non-singleton elements as clusters at threshold $I(\mathsf{Z}_V)$, and proceeds iteratively by picking any cluster (of size larger than two) and computing its finest optimal partition, etc. In our example, $\mathcal{P}^*(\mathsf{Z}_V)$ is shown in Fig. 1b, which results in the clusters $\{\{1,2,3\},\{4,5\}\}$ at threshold $I(\mathsf{Z}_V) = 0$. Next, $\mathcal{P}^*(\mathsf{Z}_{\{1,2,3\}})$ is shown in Fig. 1c, which results in the cluster $\{1,2\}$ at threshold $I(\mathsf{Z}_{\{1,2,3\}}) = 1$. After this, the divisive algorithm terminates with the entire hierarchy of clusters as in (2.9). (In this example, the algorithm did not compute the last threshold in (2.9), but it can do so by letting it proceed to clusters of size two.) □

IV. Principal sequence and minimum norm base

As discussed earlier, the PSP of h is polynomial time solvable, which characterizes the info-clustering solution by Proposition 2.1. Our agglomerative algorithm will make use of yet another polynomial time solvable structure, discussed below, that is closely related to the PSP.

Consider any submodular function $f : 2^U \to \mathbb{R}$ (2.4) on the finite ground set U. For $\lambda \in \mathbb{R}$, define

$$S_\lambda(f) := \max \operatorname*{arg\,min}_{B\subseteq U} f(B) - \lambda|B|, \tag{4.1}$$

where $\max$ is the inclusion-wise maximum. Note that the minimization in (4.1) is a *submodular function minimization (SFM)* since the function $B \subseteq U \mapsto f(B) - \lambda|B|$ is also submodular. It is well-known that the minimizers of a submodular function form a lattice w.r.t. set inclusion [23], and so the maximum in (4.1) exists and is unique.

Proposition 4.1 ([11, 13]) *$S_\lambda(f)$ for $\lambda \in \mathbb{R}$ satisfies*

$$S_{\lambda'}(f) \subseteq S_{\lambda''}(f) \quad \textit{iff} \quad \lambda' \leq \lambda''.$$

This defines a sequence, for some positive integer N',

$$S_{\lambda_0} := \emptyset \subsetneq S_{\lambda_1} \subsetneq \cdots \subsetneq S_{\lambda_{N'-1}} \subsetneq S_{\lambda_{N'}} := V \tag{4.2}$$

of distinct sets, which (together with the sequence $-\infty = \lambda_0 < \lambda_1 < \cdots < \lambda_{N'} < \lambda_{N'+1} := \infty$ of λ values at which S_λ changes) is referred to as the principal sequence (PS). □

W.l.o.g., we assume f is normalized, i.e., $f(\emptyset) = 0$, since we can redefine f as $f - f(\emptyset)$ without affecting the solutions to the SFM in (4.1), i.e., aside from a shift in the λ_i values, the PS (4.2) is invariant to a constant shift in the submodular function. The polyhedron $\mathrm{P}(f)$ and base polyhedron $\mathrm{B}(f)$ of the submodular function f are defined as

$$\mathrm{P}(f) := \{x_U \in \mathbb{R}^U \mid x(B) \leq f(B), \forall B \subseteq U\} \tag{4.3a}$$

$$\mathrm{B}(f) := \{x_U \in \mathrm{P}(f) \mid x(U) = f(U)\}, \tag{4.3b}$$

where $x_U := (x_i \mid i \in U)$ and $x(B) := \sum_{i \in B} x_i$. (N.b., $\mathrm{P}(f)$ and $\mathrm{B}(f)$ are non-empty since $f(\emptyset) \geq 0$.) With $\|x_U\|$ denoting the Euclidean norm of the vector x_U, the following holds.

Proposition 4.2 ([11, 24]) *For any normalized submodular function f, the minimization*

$$\min\{\|x_U\| \mid x_U \in \mathrm{B}(f)\} \tag{4.4}$$

has a unique solution x^*_U, called the minimum (Euclidean) norm base, which satisfies

$$x^*_i = \min\{\lambda \in \mathbb{R} \mid i \in S_\lambda(f)\} \quad \forall i \in U, \textit{or equiv.}, \tag{4.5a}$$

$$S_\lambda(f) = \{i \in U \mid x^*_i \leq \lambda\}, \qquad \forall \lambda \in \mathbb{R}, \tag{4.5b}$$

where the equivalence follows directly from Proposition 4.1. □

The minimum norm base, and so the PS by (4.5b), may be computed using Wolfe's minimum norm point algorithm as in [24]. Conversely, by (4.5a), the minimum norm base can also be computed by any SFM algorithm that solves S_λ for any λ. However, the minimum norm point algorithm was shown [24] empirically to outperform other SFM algorithms.

V. Main results

Instead of the divisive approach, we consider here an agglomerative approach, Algorithm 1, that computes γ_ℓ and $\mathcal{P}_\ell$ by iteratively producing coarser partitions from finer ones, i.e., from $\mathcal{P}_N$ to $\mathcal{P}_1$. (Note that N needs not be known

a priori.) We will give an efficient implementation (thereby resolving the inefficiency of the divisive approach) of the subroutine `agglomerate` in Algorithm 2, which computes γ_ℓ and $\mathcal{P}_{\ell-1}$ from $\mathcal{P}_\ell$ for any ℓ (thereby returning the clusters in the desired order). In particular, we will show that it suffices to compute

$$I^*(\mathsf{Z}_V) := \max\{I(\mathsf{Z}_C) \mid C \subseteq V, |C| > 1\} \quad \text{and} \tag{5.1a}$$

$$\mathcal{C}^*(\mathsf{Z}_V) := \operatorname{maximal}\{C \subseteq V \mid |C| > 1, \; I(\mathsf{Z}_C) = I^*(\mathsf{Z}_V)\}, \tag{5.1b}$$

which are, respectively, the last critical value γ_N [7, Theorem 1] and the last set of clusters (i.e., the non-singleton elements of the second last partition $\mathcal{P}_{N-1}$). Recall (2.2a) and note that if the partition $\mathcal{P} = \{\{i\} \mid i \in V\}$ is optimal then the MMI is equal to

$$\begin{aligned} J_{\mathrm{T}}(\mathsf{Z}_V) &:= I_{\{\{i\}|i\in V\}}(\mathsf{Z}_V) \\ &= \frac{1}{|V|-1}\left[\sum\nolimits_{i\in V} h(\{i\}) - h(V)\right], \end{aligned} \tag{5.2}$$

which is the normalized version (see [8] for more details) of the well-known Watanabe's total correlation [25]. The quantities I^* and $\mathcal{C}^*$ turn out to have a fundamental connection with the normalized total correlation (5.2), which leads to the desired efficient implementation.

Algorithm 1: Agglomerative info-clustering.

Data: Statistics of Z_V sufficient for calculating the entropy function $h(B)$ for $B \subseteq V := \{1, \ldots, n\}$.

Result: Two arrays L and PSP that contain the γ_ℓ's and $\mathcal{P}_\ell$'s in Proposition 2.1. More precisely, for $1 \le \ell \le N$, the entries of the arrays L and PSP are $\mathsf{L}[s] = \gamma_\ell$ and $\mathsf{PSP}[s] = \mathcal{P}_\ell$ for $s = |\mathcal{P}_\ell|$, and are null otherwise.

L, $\mathsf{PSP} \leftarrow$ empty arrays, each of length n;
$\mathsf{PSP}[n] \leftarrow \{\{i\} : i \in V\}$, $s \leftarrow n$;
while $s > 1$ **do**
    $(\gamma, \mathcal{P}') \leftarrow$ `agglomerate`$(\mathsf{PSP}[s])$;
    $\mathsf{L}[s] \leftarrow \gamma$; $s \leftarrow |\mathcal{P}'|$; $\mathsf{PSP}[s] \leftarrow \mathcal{P}'$;

The main result is the implementation of `agglomerate` in Algorithm 2 that computes the PSP, and so the info-clustering solution, iteratively from finer partitions to coarser ones. This is done by computing the PS using a subroutine `min_norm_base` that computes the minimum norm base for (4.4). An explicit implementation of `min_norm_base` can be found in [24] using Wolfe's minimum norm point algorithm.

To explain Algorithm 2, we need the following two results that 1) relate the last critical value of a set of RVs to the normalized total correlation and, consequently, to the minimum norm base; and 2) relate any critical value of a set of RVs to the last critical value of a set of agglomerated RVs. Each theorem will be illustrated with an example. The statements of the results may be skimmed first and read more carefully as we go through the examples.

Algorithm 2: Implementation of `agglomerate`.

Data: $\mathcal{P}$ is equal to $\mathcal{P}_\ell$ for some $1 \le \ell \le N$ in (2.6).

Result: $(\gamma, \mathcal{P}')$ is equal to $(\gamma_\ell, \mathcal{P}_{\ell-1})$.

Enumerate $\mathcal{P}$ as $\{C_1, \dots, C_k\}$ for some $k > 1$ and disjoint C_i's;

$\mathsf{x} \leftarrow$ empty array of size k;

for $j \leftarrow 1$ **to** k **do**

  Define g_j as the function $B \subseteq \{j+1, \dots, k\} \mapsto h\left(\bigcup_{i \in B \cup \{j\}} C_i\right) - \sum_{i \in B \cup \{j\}} h(C_i)$;

  $\mathsf{x}[j] \leftarrow$ `min_norm_base` $(g_j, \{j+1, \dots, k\})$;

$\gamma \leftarrow -\min_{i,j:1 \le j < i \le k} \mathsf{x}[j][i]$, $\quad \mathcal{P}' \leftarrow \emptyset$;

for $j \leftarrow 1$ **to** k **do**

  if $C_j \not\subseteq \bigcup \mathcal{P}'$ **then**

    add $C_j \cup \bigcup \{C_i \mid i \in \{j+1, \dots, k\}, \mathsf{x}[j][i] \le -\gamma\}$ to $\mathcal{P}'$;

function `min_norm_base` (f, U)**:**

  return an array x (indexed by U) that solves (4.4).

Theorem 5.1 *With $V := \{1, \dots, n\}$ and h denoting the entropy function of Z_V, let, for any $j \in V$,*

$$U_j := \{i \in V \mid i > j\} \text{ and}$$

$$g_j(B) := h(B \cup \{j\}) - \sum\nolimits_{i \in B \cup \{j\}} h(\{i\}) \quad \text{for } B \subseteq U_j,$$

which is a normalized submodular function. Then,

$$I^*(\mathsf{Z}_V) = \max_{C \subseteq V : |C| > 1} J_T(\mathsf{Z}_C) \tag{5.3a}$$

$$= -\min_{j \in V} \min_{i \in U_j} x_i^{(j)} \tag{5.3b}$$

$$\mathcal{C}^*(\mathsf{Z}_V) = \text{maximal} \operatorname*{arg\,max}_{C \subseteq V : |C| > 1} J_T(\mathsf{Z}_C) \tag{5.4a}$$

$$= \text{maximal} \left\{ \{j^*\} \cup \operatorname*{arg\,min}_{i \in U_{j^*}} x_i^{(j^*)} \,\middle|\, j^* \in \operatorname*{arg\,min}_{j \in V} \left(\min_{i \in U_j} x_i^{(j)} \right) \right\}, \tag{5.4b}$$

where $x_{U_j}^{(j)}$ is the minimum norm base for g_j (4.4). □

PROOF See Appendix B. ■

Example 3 Consider our running example (2.8) with $V = \{1, \dots, 6\}$. By brute-force search, the maximum normalized total correlation in (5.3a) is $J_T(\mathsf{Z}_{\{1,2\}}) = 2$ achieved uniquely by $C = \{1, 2\}$. Hence, (5.3a) and (5.4a) give, respectively, $I^*(\mathsf{Z}_V) = 2$ and $\mathcal{C}^*(\mathsf{Z}_V) = \{\{1, 2\}\}$, which correspond to the last critical value and last set of clusters in (2.9), as desired by the definitions of I^* and $\mathcal{C}^*$ in (5.1).

To illustrate (5.3b) and (5.4b), the minimum norm base $x_{U_j}^{(j)}$ of g_j, for different values of j, is calculated as:

$$x_{U_j}^{(j)} = \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & \end{array} \\ \begin{cases} (\quad \textcircled{-2}, & -1, & -0.5, & -0.5, & 0), & \text{j=1} \\ (\quad\quad\quad\; & -1, & -0.5, & -0.5, & 0), & \text{j=2} \\ (\quad\quad\quad\; & & -0.5, & -0.5, & 0), & \text{j=3} \\ (\quad\quad\quad\; & & & -1, & 0), & \text{j=4} \\ (\quad\quad\quad\; & & & & 0), & \text{j=5.} \end{cases} \end{array}$$

Now, the minimum in (5.3b) is equal to -2 (circled above), which is achieved uniquely by $j = 1$ and $i = 2$. This gives $I^*(\mathsf{Z}_V) = -(-2) = 2$ as desired. In (5.4b), we have $j^* \in \{1\}$ and $\arg\min_{i \in U_{j^*}} x_i^{(j^*)} = \{2\}$. Hence, $\mathcal{C}^*(\mathsf{Z}_V) = \text{maximal}\{\{1\} \cup \{2\}\} = \{\{1,2\}\}$ as desired. □

We remark that (5.3a) and (5.4a) eliminate the need for minimization over partitions in calculating the MMI in (5.1a) and (5.1b). They serve as an intermediate step that leads to (5.3b) and (5.4b), thereby relating the last critical value and last set of clusters to the minimum norm base.

So far we succeeded in posing the computation of the last critical value I^* and last set of clusters $\mathcal{C}^*$ as a minimum norm base problem. However, our objective is to compute the entire clustering solution. To complete the implementation of `agglomerate`, we need the following theorem which reduces the computation of the clusters at any threshold to the computation of I^* and $\mathcal{C}^*$ for some agglomerated RVs.

Theorem 5.2 *Consider $\mathcal{P}_\ell$ and γ_ℓ as defined in Proposition 2.1 and, for $1 \le \ell \le N$, write*

$$\mathcal{P}_\ell = \{C_j \mid j \in U\} \quad \textit{and} \quad \mathsf{Z}'_j := \mathsf{Z}_{C_j} \tag{5.5}$$

for some index set U and disjoint subsets C_j for $j \in U$. Then, for $1 \le \ell \le N$, we have

$$\gamma_\ell = \max_{\mathcal{F} \subseteq \mathcal{P}_\ell : |\mathcal{F}| > 1} I(\mathsf{Z}_{\bigcup \mathcal{F}}) \tag{5.6a}$$

$$= I^*(\mathsf{Z}'_U), \tag{5.6b}$$

$$\mathcal{P}_{\ell-1} \setminus \mathcal{P}_\ell = \text{maximal}\left\{ \bigcup \mathcal{F} \,\middle|\, \mathcal{F} \in \arg\max_{\mathcal{F} \subseteq \mathcal{P}_\ell : |\mathcal{F}| > 1} I(\mathsf{Z}_{\bigcup \mathcal{F}}) \right\} \tag{5.7a}$$

$$= \left\{ \bigcup_{j \in B} C_j \mid B \in \mathcal{C}^*(\mathsf{Z}'_U) \right\}, \tag{5.7b}$$

where $\bigcup \mathcal{F} := \bigcup_{B \in \mathcal{F}} B$ for convenience. □

PROOF See Appendix B. ■

Example 4 We now illustrate Theorems 5.1 and 5.2 using our running example (2.8).

In the first iteration, $l = N$, consider (5.5) with $\mathcal{P}_\ell$ being the partition into singletons, namely, let $\mathsf{Z}'_i = \mathsf{Z}_i$ for $i = 1, \ldots, 6$ (i.e., $C_i = \{i\}$). By (5.3b) and (5.4b), we have $I^*(\mathsf{Z}_{\{1,\ldots,6\}}) = 2$ and $\mathcal{C}^*(\mathsf{Z}_{\{1,\ldots,6\}}) = \{\{1,2\}\}$. (See Example 3.) By (5.6b) and (5.7b), we have $\gamma_N = 2$ and $\mathcal{P}_{N-1} \setminus \mathcal{P}_N = \{\{1,2\}\}$ as desired. In other words, when called with the singletons partition as the input partition, the function `agglomerate` returns the threshold value γ_N and partition $\mathcal{P}_{N-1}$ correctly.

In the next iteration, $l = N-1$, consider (5.5) with $\mathcal{P}_\ell$ as computed above, namely, let $\mathsf{Z}'_1 = \mathsf{Z}_{\{1,2\}}$ and $\mathsf{Z}'_i = \mathsf{Z}_{i+1}$ for $i = 2, \ldots, 5$ (i.e., $C_1 = \{1,2\}$ and $C_i = \{i+1\}$). The minimum norm base $x^{(j)}_{U_j}$ of g_j (defined using the entropy function of $\mathsf{Z}'_{\{1,\ldots,5\}}$) is

$$x^{(j)}_{U_j} = \begin{array}{c} \begin{array}{ccccccc} & 1 & 2 & 3 & 4 & 5 & \\ \end{array} \\ \begin{cases} (& \textcircled{-1}, & \text{-0.5}, & \text{-0.5}, & 0), & \text{j=1} \\ (& & \text{-0.5}, & \text{-0.5}, & 0), & \text{j=2} \\ (& & & \textcircled{-1}, & 0), & \text{j=3} \\ (& & & & 0), & \text{j=4} \end{cases} \end{array}$$

By (5.3b) and (5.4b), we have $I^*(\mathsf{Z}'_{\{1,\ldots,5\}}) = 1$ and $\mathcal{C}^*(\mathsf{Z}'_{\{1,\ldots,5\}}) = \{\{1,2\},\{3,4\}\}$, which in terms of Z_V, results in the clusters $\{\{1,2,3\},\{4,5\}\}$. By (5.6b) and (5.7b), we have $\gamma_{N-1} = 1$ and $\mathcal{P}_{N-2} \backslash \mathcal{P}_{N-1} = \{\{1,2,3\},\{4,5\}\}$ as desired. In other words, when called with the input partition $\{\{1,2\},\{3\},\{4\},\{5\},\{6\}\}$, the function `agglomerate` returns the threshold value γ_{N-1} and partition $\mathcal{P}_{N-2}$ correctly.

In the next iteration, $l = N-2$, consider (5.5) with $\mathcal{P}_\ell$ as computed above, namely, let $\mathsf{Z}'_1 = \mathsf{Z}_{\{1,2,3\}}$ ($C_1 = \{1,2,3\}$), $\mathsf{Z}'_2 = \mathsf{Z}_{\{4,5\}}$ ($C_2 = \{4,5\}$), and $\mathsf{Z}'_3 = \mathsf{Z}_6$ ($C_3 = \{6\}$). The minimum norm base $x^{(j)}_{U_j}$ of g_j (defined using the entropy function of $\mathsf{Z}'_{\{1,2,3\}}$) is given as

$$x^{(j)}_{U_j} = \begin{array}{c} \begin{array}{ccccc} & 1 & 2 & 3 & \\ \end{array} \\ \begin{cases} (& \textcircled{0}, & \textcircled{0}), & \text{j=1} \\ (& & \textcircled{0}), & \text{j=2} \end{cases} \end{array}$$

By (5.3b) and (5.4b), we have $I^*(\mathsf{Z}'_{\{1,2,3\}}) = 0$ and $\mathcal{C}^*(\mathsf{Z}'_{\{1,2,3\}}) = \{1,2,3\}$, which in terms of Z_V, results in the trivial cluster $\{1,\ldots,6\}$. By (5.6b) and (5.7b), $\gamma_{N-2} = 0$ and $\mathcal{P}_{N-3} \backslash \mathcal{P}_{N-2} = \{1,\ldots,6\}$ as desired. At this point the agglomerative algorithm terminates. □

The complexity of `agglomerate` (Algorithm 2) is mainly due to the computation of the minimum norm base in line 5. This computation is repeated at most n times. With $\mathrm{MNB}(n)$ denoting the complexity of the minimum norm base algorithm for a ground set of size n, `agglomerate` runs in time $O(n\,\mathrm{MNB}(n))$. Since the agglomerative info-clustering algorithm in Algorithm 1 invokes function `agglomerate` $N-1 \leq n-1$ times, it runs in time $O(n^2\,\mathrm{MNB}(n))$, which is equivalent to that of [7, Algorithm 3], assuming that the submodular function minimization therein is implemented by the minimum norm base algorithm, i.e., with $\mathrm{SFM} = \mathrm{MNB}$. In contrast, the divisive info-clustering algorithm in [7, Algorithm 2] makes $N-1$ calls to a subroutine that calculates the fundamental partition. However, computing the fundamental partition appears to take time $O(n^2\,\mathrm{MNB}(n))$, which leads to an overall complexity of $O(n^3\,\mathrm{MNB}(n))$ for the divisive clustering.

VI. Data Structure for Representing Hierarchy of Clusters

In this section we describe a data structure that can record the solution of an agglomerative clustering algorithm efficiently, namely, with linear space complexity, by providing the following function:

- `merge(`i,j,γ`)` runs in $O(\log n)$ time and merges nodes $i, j \in V$ into the same cluster for all threshold values $< \gamma$.

An agglomerative clustering algorithm should call `merge` successively with distinct pairs of nodes and non-increasing values of γ. After all the nodes are merged into the trivial cluster V, the following functions can be used to retrieve information about the clusters in linear/sub-linear time:

- `getCriticalValues()` runs in $O(n)$ time and returns the ordered list of critical threshold values γ_ℓ's in (2.6) where the set of clusters changes.
- `getPartition(`γ`)` runs in $O(n)$ time and returns the partition $\mathcal{P}$ of V such that $\mathcal{C}_\gamma(\mathsf{Z}_V) = \mathcal{P} \backslash \{\{i\} \mid i \in V\}$. In other words, $\mathcal{P}$ is the partition $\mathcal{P}_\ell$ in (2.6) with the smallest $\gamma_\ell \geq \gamma$.
- `similarity(`i,j`)` runs in $O(\log n)$ time and returns the critical value at which nodes $i, j \in V$ no longer belong to the same cluster, i.e., the largest critical value γ_ℓ s.t. i and j belong to the same cluster for all thresholds $\gamma' < \gamma_\ell$.

Internally, the hierarchy of clusters is represented by a weighted rooted tree on V, in which each node $i \in V$ is associated with the following data:

- parent[i]: The parent of i. If i has no parent, then parent[i]$= i$.
- children[i]: The list of children of i. If i has no children, then the list is empty.
- weight[i]: The weight of the edge between i and its parent. It is initialized to $-\infty$.
- rank[i]: The depth of the maximal subtree rooted at i. It is initialized to 0.

Fig. 2 gives the pseudocode of all the functions that support the desired data structure. The idea is to represent clusters by a weighted tree T where each edge $\{i, j\} \in \mathcal{E}(T)$ has certain real-valued weight.

Definition 6.1 Given a tree T, the set of clusters at threshold γ is the set $\mathcal{P}(T_\gamma) \setminus \{\{i\} \mid i \in V\}$ of connected components of the forest T_γ obtained from T by removing edges of weights $\leq \gamma$. □

The function `initialize` constructs the leaves, namely, the singleton subsets $\{i\}$ for $i \in V$. To construct T starting from its leaves, we call the function `merge(`i, j, γ`)` for some i, j, γ, which verifies if i and j are already connected in T, and if not, adds an edge with weight γ between any node i' connected to i and any node j' connected to j. (A more specific choice of i' and j' will be discussed shortly.) The computational complexity is $O(\log n)$ if the verification of connectedness can be done in $O(\log n)$ time. This can be done using the same idea as the disjoint-set forest with union by rank [26, 27], by providing the function:

- `find(`i`)` that runs in $O(\log n)$ time and returns a representative of the component containing i.

Then, node i is connected to node j iff `find(`i`)=find(`j`)`. To implement `find`, with the desired $O(\log n)$ complexity, the partial clustering solution is represented by a rooted forest where each connected component is a rooted tree and so `find` can simply return the root of the tree as the representative of the component. To maintain the invariance that the graph is a rooted forest, `merge(`i, j, γ`)` adds a edge with weight γ between the root i' of i and the root j' of j if $i' \neq j'$. To ensure $O(\log n)$ complexity for `find` and therefore `merge`, the root of the deeper tree is chosen to be the parent of the other root since this ensures that the depth of any tree in the forest to be $O(\log n)$ at all times. Note that, different from [28], the path compression technique for disjoint-set forest cannot be adopted here as it would destroy the hierarchical structure.

```
function initialize():
    for i ∈ V do
        parent[i]← i, children[i]←empty list,
         weight[i]← −∞, rank[i]← 0;

function find(i):
    if parent[i]=i then return i;
    else return find(parent[i]);

function merge(i,j,γ):
    i′ ←find(i), j′ ←find(j);
    if i′=j′ then return;
    if rank[i′] ≤ rank[j′] then
        parent[i′]← j′;
        add i′ to children[j′];
        weight[i′]← γ;
        if rank[i′]=rank[j′] then
            rank[j′]←rank[j′]+1;
    else
        parent[j′]← i′;
        add j′ to children[i′];
        weight[j′]← γ;
```

```
function getCriticalValues():
    return weight;

function getPartition(γ):
    S ← V, P ← ∅;
    while S ≠ ∅ do
        i ← any node from S;
        B ← set of nodes reachable from i, by
            going from a reachable node j to
            parent[j] if weight[j]> γ and to
            j′ ∈children[j] if weight[j′]> γ;
        S ← S \ B, add B to P;

function similarity(i,j):
    if i = j then return ∞;
    i′ ←parent[i], j′ ←parent[j],
     γi ←weight[i], γj ←weight[j];
    if i′ = j then return γi;
    if j′ = i then return γj;
    if γi ≥ γj then
        return similarity(i′,j);
    else return similarity(i,j′);
```

Fig. 2: Pseudocode to construct (left) and query (right) the data structure for the clustering solution.

Finally, using the data structure described above, the agglomerative info-clustering algorithm in Algorithm 1 and 2 can be implemented as shown in Algorithm 3.

The following example demonstrates Algorithm 3 and the data structure in this section.

Example 5 Consider our running example (2.8) with $V = \{1, \ldots, 6\}$. In Algorithm 3, the function `initialize()` results in the empty forest in Fig. 3a, where each node is a root, i.e., $\mathsf{parent}[i] = i$ for $i \in V$. (A root is highlighted in gray.) After initialization, the function `getPartition(`$-\infty$`)` returns the singletons partition $\mathcal{P} = \{\{i\} \mid i \in V\}$ of size $k = 6$ since initially $\mathsf{weight}[i] = -\infty$, for $i \in V$, and so the only node reachable from i is node i itself. Subsequently, the algorithm computes the minimum norm base $\mathsf{x}[j]$, whose minimum entry is given as $\mathsf{x}[j][i] = -2$ for $j \in C_1 = \{1\}, i \in C_2 = \{2\}$, as we saw in Example 3, resulting in $\gamma = 2$. Now, the algorithm updates the tree by calling `merge(`$2, 1, 2$`)`, which sets $\mathsf{parent}[2] = 1$, $\mathsf{weight}[2] = 2$, $\mathsf{children}[1] = \{2\}$, $\mathsf{rank}[1] = 1$, and leaves

Algorithm 3: Implementation of Agglomerative info-clustering using an efficient data structure.

Data: Statistics of Z_V sufficient for calculating the entropy function $h(B)$ for $B \subseteq V := \{1, \ldots, n\}$.

Result: Info-clustering solution in Proposition 2.1 recorded by parent, children, weight and rank.

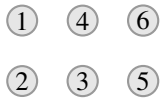
```
initialize();
𝒫 ←getPartition(−∞);k ← |𝒫|;
while k > 1 do
    x← empty array of size k;
    for j ← 1 to k do
        Define g_j as the function B ⊆ {j+1,...,k} ↦ h(⋃_{i∈B∪{j}} C_i) − Σ_{i∈B∪{j}} h(C_i);
        x[j]←min_norm_base (g_j, {j+1,...,k});
    γ ← − min_{i,j:1≤j<i≤k} x[j][i];
    for j ← 1 to k do
        for i ← 1 to j − 1 do
            if x[j][i]≤−γ then
                merge(u,v,γ) for an arbitrary u ∈ C_i and an arbitrary v ∈ C_j;
    𝒫 ←getPartition(−∞);k ← |𝒫|;
```

all other variables unchanged. See Fig. 3b.

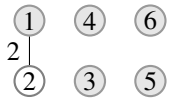
Next, the function `getPartition(`$-\infty$`)` returns the partition $\mathcal{P} = \{\{1,2\},\{3\},\{4\},\{5\},\{6\}\}$ (corresponding to the connected components in 3b) of size $k = 5$. Subsequently, the algorithm computes the minimum norm base $\mathsf{x}[j]$, whose minimum entry is given as $\mathsf{x}[j][i] = -1$ for $j \in C_1 = \{1,2\}$ and $i \in C_2 = \{3\}$, and for $j \in C_3 = \{4\}$ and $i \in C_4 = \{5\}$, as we saw in Example 4, resulting in $\gamma = 1$. Now, the algorithm updates the tree by calling `merge(`$3,2,1$`)`, which sets $\mathsf{parent}[3] = 1$, $\mathsf{weight}[3] = 1$, $\mathsf{children}[1] = \{2,3\}$, and leaves all other variables unchanged. (Equivalently, one may call `merge(`$3,1,1$`)`, which leads to the same result.) The algorithm then calls `merge(`$5,4,1$`)`, which sets $\mathsf{parent}[5] = 4$, $\mathsf{weight}[5] = 1$, $\mathsf{children}[4] = \{5\}$, $\mathsf{rank}[4] = 1$, and leaves all other variables unchanged. See Fig. 3c.

Next, the function `getPartition(`$-\infty$`)` returns the partition $\mathcal{P} = \{\{1,2,3\},\{4,5\},\{6\}\}$ (corresponding to the connected components in 3c) of size $k = 3$. Subsequently, the algorithm computes the minimum norm base $\mathsf{x}[j]$, whose minimum entry is given as $\mathsf{x}[j][i] = 0$ for $j \in C_1 = \{1,2,3\}$ and $i \in C_2 \cup C_3 = \{4,5,6\}$, and for $j \in C_2 = \{4,5\}$ and $i \in C_3 = \{6\}$, as we saw in Example 4, resulting in $\gamma = 0$. Now, the algorithm updates the tree by calling `merge(`$4,1,0$`)`, which sets $\mathsf{parent}[4] = 1$, $\mathsf{weight}[4] = 0$, $\mathsf{children}[1] = \{2,3,4\}$, $\mathsf{rank}[1] = 2$ and leaves all other variables unchanged. The algorithm then calls `merge(`$4,6,0$`)`, which sets $\mathsf{parent}[6] = 1$, $\mathsf{weight}[6] = 0$, $\mathsf{children}[1] = \{2,3,4,6\}$, and leaves all other variables unchanged. See Fig. 3d.

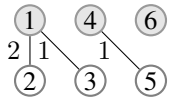
Finally, the algorithm terminates upon calling the function `getPartition(`$-\infty$`)` which returns the partition



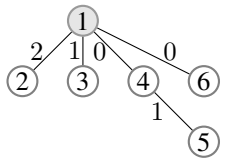
(a) Initialization.



(b) merge $(2, 1, 2)$.



(c) merge $(3, 2, 1)$, then merge $(5, 4, 1)$.



(d) merge $(4, 1, 0)$, then merge $(4, 6, 0)$.

Fig. 3: Construction of a rooted tree that represents the hierarchical clustering solution, where a root node is highlighted in gray. (a) Initially, we start with an empty forest without edges where each node is a root. (b), (c), and (d) show different stages of building the tree according to Algorithm 3. (d) is the final tree representing the clustering solution. Now, for any γ, the clustering solution can be obtained from the tree in (d) according to Definition 6.1, i.e., by removing all edges of weight $\leq \gamma$ and returning the connected components as the clustering solution.

$\mathcal{P} = \{V\}$ of size $k = 1$. □

VII. Conclusion

We have described an agglomerative info-clustering approach that merges smaller clusters into larger clusters successively. In particular, it reveals a fundamental property of the MMI, namely, that the MMI can be computed by maximizing the normalized total correlation successively over subsets of agglomerated RVs. While faster info-clustering algorithms are possible as in the case of the Markov tree model [7] and the graphical model [29, 30], the current agglomerative approach gives the fastest algorithm without assuming any special source models. To further improve the performance, heuristics such as [31] may be adapted to improve the minimum norm base algorithm.

Appendix A
Principal Squence of Partitions and Dilworth Truncation

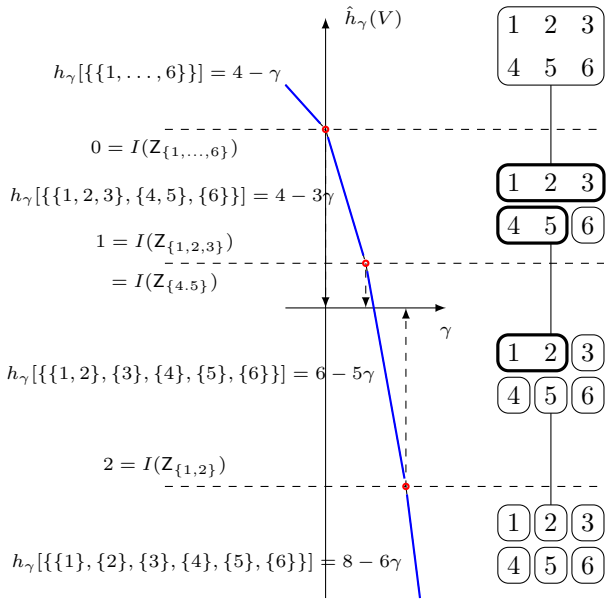
In this section we give a brief discussion on the PSP and the Dilworth truncation, which will be usefull for proving the main results in Appendix B. For any real number $\gamma \in \mathbb{R}$, define the *residual entropy function* [8] of a random vector Z_V as

$$h_\gamma(B) := h(B) - \gamma \qquad \text{for } B \subseteq V, \tag{A.1}$$

where h is the entropy function (2.3). The residual entropy function is also submodular and its *Dilworth truncation* evaluated at V is defined as [23]

$$\hat{h}_\gamma(V) := \min_{\mathcal{P} \in \Pi(V)} h_\gamma[\mathcal{P}] \quad \text{where} \tag{A.2a}$$

$$h_\gamma[\mathcal{P}] := \sum\nolimits_{C \in \mathcal{P}} h_\gamma(C). \tag{A.2b}$$



(a) $\hat{h}_\gamma(V)$ (b) PSP

Fig. 4: Dilworth truncation

This definition provides an alternative charaterization of the MMI as the smallest γ such that (A.2a) holds for some $\mathcal{P} \in \Pi'(V)$ [8]. Morevoer, we have the following structural property of optimal partitions to (A.2a).

Proposition A.1 ([9]) *Submodularity* (2.4) *of* h (2.3) *implies that the set of optimal partitions to the Dilworth truncation* (A.2a) *forms a lattice, called the Dilworth truncation lattice, with respect to the partial order* (2.1). *Furthermore, if* $\mathcal{P}'$ *and* $\mathcal{P}''$ *are the optimal partitions for* γ' *and* γ'' *respectively, then* $\gamma' < \gamma''$ *implies* $\mathcal{P}' \succeq \mathcal{P}''$. □

In particular, the minimum/finest optimal partition exists and characterizes the info-clustering solution as follows:

Proposition A.2 ([7, Corollary 2]) *For a finite set* V *with size* $|V| > 1$ *and a random vector* Z_V,

$$\mathcal{C}_\gamma(\mathsf{Z}_V) = \Big[\min\{\mathcal{P} \in \Pi(V) \mid h_\gamma[\mathcal{P}] = \hat{h}_\gamma(V)\}\Big] \setminus \{\{i\} \mid i \in V\}\,, \tag{A.3}$$

namely, the non-singleton elements of the finest optimal partition to the Dilworth truncation (A.2a). □

Hence, in Proposition 2.1, the sequence of $\mathcal{P}_\ell$ for ℓ from 1 to N with the corresponding γ_ℓ also characterizes the minimum optimal partitions to (A.2a) for all $\gamma \in \mathbb{R}$. This sequece of partitions (together with the critical values) is known as the *principal sequence of partitions* (PSP) of h and was introduced in [9]. In other words, given the PSP of the entropy function, one can readily obtain the clustering solution, and vice versa.

The following example demonstrates the Dilworth truncation for the RVs Z_V in Example 1, where the PSP in Fig. 1a is shown again in Fig. 4b for convenience.

Example 6 For $\mathsf{Z}_{\{1,\ldots,6\}}$ in (2.8), the Dillworth truncation $\hat{h}_\gamma(V)$ (A.2a) is shown in Fig. 4a. The Dillowrth truncation is piecewise linear with at most $|V|$ turning points. A turning point $p_i := (\gamma_i, \hat{h}_{\gamma_i}(V))$ occurs when (A.2a) has more than one solution. The collection of such optimal solutions is the Dillowrth truncation lattice at γ_i indicated in Proposition A.1.[1] The finest optimal partition at γ_i remains (while all other partitions seieze to be) optimal until the next turning point. In other words, the finest optimal partition at γ_i determines the line segment that follows γ_i. The sequence of the finest optimal partitions is the PSP, whch is shown in Fig. 4b. □

APPENDIX B
PROOFS OF MAIN RESULTS

PROOF (THEOREM 5.1) Consider $\gamma_N, \mathcal{P}_{N-1}$, and $\mathcal{P}_N$ as in Proposition 2.1 and let h_γ be as in (A.1). By Proposition A.2, as detailed below, we have for all $C \in \mathcal{P}_{N-1} \setminus \mathcal{P}_N$ that

$$h_{\gamma_N}(C) = \sum_{i\in C} h_{\gamma_N}(\{i\}) \quad \text{or equivalently,} \tag{B.1a}$$

$$\gamma_N = J_T(\mathsf{Z}_C). \tag{B.1b}$$

- The equivalence between (B.1a) and (B.1b) follows immediately from the definition of J_{T} (5.2). More precisely, (B.1b) is equivalent to

$$\begin{aligned}
&\gamma_N = \frac{1}{|C|-1}\left[\sum_{i\in C} h(\{i\}) - h(C)\right] && \text{by (5.2)}\\
\iff\ &(|C|-1)\gamma_N = \sum_{i\in C} h(\{i\}) - h(C) && \because |C| > 1\\
\iff\ &h_{\gamma_N}(C) = \sum_{i\in C} h_{\gamma_N}(\{i\}) && \text{by (A.1)}
\end{aligned}$$

  which is equivalent to (B.1a) as desired.

- "$\geq$" for (B.1a) follows from

$$h_{\gamma_N}\left[\{C\} \cup \{\{i\} \mid i \in V \setminus C\}\right] \geq h_{\gamma_N}[\mathcal{P}_N]$$

  since $\mathcal{P}_N$ is optimal to $\hat{h}_{\gamma_N}(V)$ (A.2) by construction. (The notation $h_{\gamma_N}[\cdot]$ is defined in (A.2b).)

- To explain "$\leq$" for (B.1a), note that $\mathcal{P}_{N-1}$ is an optimal partition to the Dilworth truncation (A.2a) for $\gamma \in [\gamma_{N-1}, \gamma_N)$ by Proposition A.2. By continuity of $h_\gamma[\mathcal{P}_{N-1}]$ (A.2b) with respect to γ, we have that $\mathcal{P}_{N-1}$ is also optimal for $\gamma = \gamma_N$, i.e.,

$$h_{\gamma_N}[\mathcal{P}_N] = h_{\gamma_N}[\mathcal{P}_{N-1}], \tag{B.2}$$

  by the optimality of $\mathcal{P}_N$. Hence, since

$$\sum_{i\in\mathcal{P}_N} h_{\gamma_N}(\{i\}) = \sum_{C\in\mathcal{P}_{N-1}} \sum_{i\in C} h_{\gamma_N}(\{i\}),$$

  we have,

$$\sum_{C\in\mathcal{P}_{N-1}\setminus\mathcal{P}_N} \left[h_{\gamma_N}(C) - \sum_{i\in C} h_{\gamma_N}(\{i\})\right] = 0.$$

[1]The Dillowrth truncation lattice is not shown in Fig. 4. This lattice is shown in Figs. 1b and 1c for Z_V and $\mathsf{Z}_{\{1,2,3\}}$, respectively.

Each term in the bracket is non-negative as argued before, and so must be equal to zero.

Consider any maximal solution C' to the r.h.s. of (5.3a). Then,

$$\gamma_N \le J_{\mathrm{T}}(\mathsf{Z}_{C'}) \quad \text{or equivalently,} \tag{B.3a}$$

$$h_{\gamma_N}(C') \le \sum_{i\in C'} h_{\gamma_N}(\{i\}), \tag{B.3b}$$

which follow from (B.1b) and (B.1a), respectively, since (5.3a) does not require $C' \in \mathcal{P}_{N-1} \setminus \mathcal{P}_N$. With

$$\mathcal{P}' := \{C'\} \cup \{\{i\} \mid i \in V \setminus C'\},$$

(B.3b) implies

$$h_{\gamma_N}[\mathcal{P}'] \le h_{\gamma_N}[\mathcal{P}_N],$$

which must hold with equality by the optimality of $\mathcal{P}_N$. Therefore, (B.3a) also holds with equality, and so we have (5.3a) since $\gamma_N = I^*(\mathsf{Z}_V)$. Note that, by (B.2), $\mathcal{P}_{N-1}$ is also an optimal partition. Furthermore, it is the largest/coarsest such partition since it is optimal for some $\gamma < \gamma_N$ and therefore larger/coarser than all optimal partitions for $\gamma = \gamma_N$ by Proposition A.1. Therefore, the set of maximal optimal solutions to the r.h.s. of (5.3a) is $\mathcal{P}_{N-1} \setminus \mathcal{P}_N$, which is $\mathcal{C}^*(\mathsf{Z}_V)$ trivially, and so we have (5.4a).

To prove (5.3b) and (5.4b), let $\gamma^* = I^*(\mathsf{Z}_V)$. Then, it follows from (5.3a) that

$$\begin{aligned} 0 &= \min_{C\subseteq V:|C|>1} \gamma^* - J_{\mathrm{T}}(\mathsf{Z}_C) \\ &\overset{\text{(a)}}{=} \min_{C\subseteq V:|C|>1} (|C|-1)\gamma^* + h(C) - \sum_{i\in C} h(\{i\}) \\ &\overset{\text{(b)}}{=} \min_{j\in V} \min_{B\subseteq U_j:|B|\ge 1} g_j(B) + \gamma^*|B|, \end{aligned} \tag{B.4}$$

where

- (a) is obtained by the definition of J_{T} (5.2) and by multiplying both sides of the equality by $|C|-1>0$, which does neither violate the equality nor change the set of solutions; and
- (b) is obtained by changing the variable C to (j, B) using the bijection that sets

  $$j := \min_{i\in C} i \quad \text{and} \quad B := C \setminus \{j\},$$

  which is possible because $|C|>1$. We have also applied $|B| = |C|-1$ and the definition of g_j to rewrite the expression in the minimization.

Consider any optimal solution j^* to the r.h.s. of (B.4). Then, it follows that $S_\lambda(g_{j^*})$ in (4.1) is such that

$$\begin{aligned} S_\lambda(g_{j^*}) &\overset{\text{(c)}}{\neq} \emptyset, \quad \lambda = -\gamma^* \\ S_\lambda(g_{j^*}) &\overset{\text{(d)}}{=} \emptyset, \quad \lambda < -\gamma^*, \end{aligned}$$

where

- (c) is because, by (B.4),

$$\begin{aligned}\emptyset \neq \max \operatorname*{arg\,min}_{B\subseteq U_{j^*}:|B|\geq 1} g_{j^*}(B)+\gamma^*|B| \\ = \max \operatorname*{arg\,min}_{B\subseteq U_{j^*}} g_{j^*}(B)+\gamma^*|B| \\ = S_{-\gamma^*}(g_{j^*}).\end{aligned}$$

  The last equality is by the definition (4.1) of $S_{-\gamma^*}$. The first equality is because allowing $B=\emptyset$ does not change the minimum value 0, since

$$g_{j^*}(\emptyset)+\gamma^*|\emptyset| = 0$$

  as $g_{j^*}(\emptyset)$. Doing so also does not affect the maximum minimizer since the new optimal solution introduced, namely $\emptyset$, cannot be maximum trivially.

- To explain (d), note that (B.4) implies for all $B\subseteq U_{j^*}:|B|>1$ that

$$\begin{aligned} g_{j^*}(B)+\gamma^*|B| &\geq 0 && \text{and so} \\ g_{j^*}(B)-\lambda|B| &\geq (-\lambda-\gamma^*)|B| && \forall \lambda\in\mathbb{R} \\ &> 0 && \forall \lambda < -\gamma^*.\end{aligned}$$

  In other words, for $\lambda<-\gamma^*$, we have that $\emptyset$ is the unique solution to $\min_{B\subseteq U_{j^*}} g_{j^*}(B)+\gamma^*|B|$, which implies (d) by the definition (4.1) of S_λ.

Now, (c) and (d) imply that

$$\begin{aligned}-\gamma^* &= \sup\{\lambda\in\mathbb{R} \mid S_\lambda(g_{j^*})=\emptyset\} \\ &= \min_{i\in U_{j^*}} \min_{\lambda\in\mathbb{R}: i\in S_\lambda(g_{j^*})} \lambda \\ &= \min_{i\in U_{j^*}} x_i^{(j^*)} \end{aligned} \tag{B.5}$$

where the last equality is by (4.5a) since $x^{(j^*)}_{U_{j^*}}$ denotes the minimum norm base for g_{j^*}. The equality implies (5.3b) as desired.

To prove (5.4b), consider any set C^* that belongs to the r.h.s. of (5.4a). Applying the bijection

$$j^* := \min_{i\in C^*} i \quad \text{and} \quad B^* := C^*\setminus\{j^*\},$$

(j^*, B^*) is a solution to the r.h.s. of (B.4). By the inclusion-wise maximality of C^*, the set B^* is also a maximal (the maximum) solution, i.e.,

$$\begin{aligned} B^* &\overset{\text{(e)}}{=} S_{-\gamma^*}(g_{j^*}) \\ &= \{i\in U_{j^*} \mid i\in S_{-\gamma^*}(g_{j^*})\} \\ &\overset{\text{(f)}}{=} \{i\in U_{j^*} \mid x_i^{(j^*)} \leq -\gamma^*\} \\ &\overset{\text{(j)}}{=} \operatorname*{arg\,min}_{i\in U_{j^*}} x_i^{(j^*)}, \end{aligned}$$

where (e) is by (4.1); (f) is by (4.5b); and (j) is by (B.5). Hence, C^* also belongs to the r.h.s. of (5.4b).

Conversely, if (j^*, B^*) is an optimal solution to the r.h.s. of (B.4), it can be argued easily that $C^* := \{j\}^* \cup B^*$ is also a solution to the r.h.s. of (5.3a). The maximal such C^* therefore belongs to the l.h.s. of (5.4b) as desired. This completes the proof. ■

PROOF (THEOREM 5.2) First, we prove (5.6a) and (5.7a) using the general property (2.7) instead of the precise definition (2.2) of the MMI.

We first argue that, for any feasible solution $\mathcal{F}$ to the r.h.s. of (5.6a),

$$I(\mathsf{Z}_{\bigcup \mathcal{F}}) \leq \gamma_\ell. \tag{B.6}$$

Suppose to the contrary that $I(\mathsf{Z}_{\bigcup \mathcal{F}}) > \gamma_\ell$. Then, there exists $C \supseteq \bigcup \mathcal{F}$ such that $C \in \mathcal{C}_{\gamma_\ell}(\mathsf{Z}_V)$ by the definition (2.5) of clusters. However, $\mathcal{C}_{\gamma_\ell}(\mathsf{Z}_V) \subseteq \mathcal{P}_\ell$ by Proposition 2.1, which contradicts the fact that $\mathcal{F} \subseteq \mathcal{P}_\ell$ with $|\mathcal{F}| > 1$.

Next, we show that (B.6) can be achieved with equality for some feasible solution $\mathcal{F}$. Consider any $C \in \mathcal{P}_{\ell-1} \setminus \mathcal{P}_\ell$. (Such a C exists since $\mathcal{P}_\ell$ is strictly finer than $\mathcal{P}_{\ell-1}$.) Then, we have

$$C = \bigcup \mathcal{F} \text{ for some } \mathcal{F} \subseteq \mathcal{P}_\ell : |\mathcal{F}| > 1, \tag{B.7}$$

i.e., for some feasible solution $\mathcal{F}$. By Proposition 2.1, we have

$$C \in \mathcal{C}_\gamma(\mathsf{Z}_V) \quad \text{ for all } \gamma \in [\gamma_{\ell-1}, \gamma_\ell),$$

and so $I(\mathsf{Z}_C) \geq \gamma_\ell$, i.e., larger than all values in the interval. The reverse inequality also holds by (B.6) and (B.7). Hence, we have

$$I(\mathsf{Z}_C) = \gamma_\ell, \tag{B.8}$$

which implies (5.6a) as desired.

Now, we argue that the above construction gives all the optimal solutions to the r.h.s. of (5.6a), hence establishing (5.7a). For any $C \in \mathcal{P}_{\ell-1} \setminus \mathcal{P}_\ell$, (B.7) and (B.8) imply that "$\subseteq$" holds for (5.7a), because the fact that $C \in \mathcal{C}_{\gamma_{\ell-1}}(\mathsf{Z}_V)$ (by Proposition 2.1) means that it is maximal by the definition (2.5) of clusters.

To argue the reverse inclusion "$\supseteq$", consider any $\mathcal{F}$ belonging to the r.h.s. of (5.7a). By (5.6a),

$$I(\mathsf{Z}_{\bigcup \mathcal{F}}) = \gamma_\ell > \gamma_{\ell-1}$$

and so $\bigcup \mathcal{F} \in \mathcal{C}_{\gamma_{\ell-1}}$ by the definition (2.5) of clusters and the maximality of $\bigcup \mathcal{F}$. This completes the poof of (5.7a).

Consider proving (5.6b) and (5.7b). For any optimal solution $\mathcal{F}$ to (5.6a), we have

$$\mathcal{P}^*(\mathsf{Z}_{\bigcup \mathcal{F}}) = \mathcal{F} = \{C_j \mid j \in B\}$$

for some $B \subseteq U$, where the first equality is by Proposition 3.1 since $I(\mathsf{Z}_C) > \gamma_\ell$ for all $C \in \mathcal{F}$ such that $C > 1$. Hence,

$$\begin{aligned} I(\mathsf{Z}_{\bigcup \mathcal{F}}) = I_{\mathcal{P}^*(\mathsf{Z}_{\bigcup \mathcal{F}})}(\mathsf{Z}_{\bigcup \mathcal{F}}) &= I_{\mathcal{F}}(\mathsf{Z}_{\bigcup \mathcal{F}}) \\ &= I_{\{\{j\} \mid j \in B\}}(\mathsf{Z}'_B) \geq I(\mathsf{Z}'_B) \end{aligned}$$

where the inequality follows from the fact that partition $\{\{j\} \mid j \in B\}$ into singletons may not be the optimal partition of B for $I(\mathsf{Z}'_B)$. The reverse inequality also holds because, for all $\mathcal{P}' \in \Pi'(B)$, define

$$\mathcal{P} := \left\{ \bigcup_{j \in C'} C_j \mid C' \in \mathcal{P}' \right\} \in \Pi'\Big(\bigcup \mathcal{F}\Big),$$

we have $I_{\mathcal{P}'}(\mathsf{Z}'_B) = I_{\mathcal{P}}(\mathsf{Z}_{\bigcup \mathcal{F}}) \geq I(\mathsf{Z}_{\bigcup \mathcal{F}})$. Here, the inequality follows from the fact that $\mathcal{P}$ may not be the optimal partition of $\mathsf{Z}_{\bigcup \mathcal{F}}$. This completes the proof of Theorem 5.2. ∎

REFERENCES

[1] C. Chan, A. Al-Bashabsheh, and Q. Zhou, "Agglomerative info-clustering," in *2018 IEEE International Symposium on Information Theory (ISIT)*, June 2018, pp. 1725–1729.

[2] A. K. Jain, R. C. Dubes *et al.*, *Algorithms for clustering data*. Prentice hall Englewood Cliffs, 1988, vol. 6.

[3] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[4] R. Sibson, "Slink: an optimally efficient algorithm for the single-link cluster method," *The computer journal*, vol. 16, no. 1, pp. 30–34, 1973.

[5] D. Defays, "An efficient algorithm for a complete link method," *The Computer Journal*, vol. 20, no. 4, pp. 364–366, 1977.

[6] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[7] C. Chan, A. Al-Bashabsheh, Q. Zhou, T. Kaced, and T. Liu, "Info-clustering: A mathematical theory for data clustering," *IEEE Trans. on Mol. Biol. Multi-Scale Commun.*, vol. 2, no. 1, pp. 64–91, June 2016.

[8] C. Chan, A. Al-Bashabsheh, J. Ebrahimi, T. Kaced, and T. Liu, "Multivariate mutual information inspired by secret-key agreement," *Proceedings of the IEEE*, vol. 103, no. 10, pp. 1883–1913, Oct 2015.

[9] H. Narayanan, "The principal lattice of partitions of a submodular function," *Linear Algebra and its Applications*, vol. 144, no. 0, pp. 179 – 216, 1990.

[10] R. G. James, C. J. Ellison, and J. P. Crutchfield, "dit: a Python package for discrete information theory," *The Journal of Open Source Software*, vol. 3, no. 25, p. 738, 2018.

[11] S. Fujishige, "Lexicographically optimal base of a polymatroid with respect to a weight vector," *Mathematics of Operations Research*, vol. 5, no. 2, pp. 186–196, 1980.

[12] ——, *Submodular functions and optimization*, 2nd ed. Elsevier, 2005.

[13] ——, "Theory of principal partitions revisited," in *Research Trends in Combinatorial Optimization*. Springer, 2009, pp. 127–162.

[14] L. Kozachenko and N. N. Leonenko, "Sample estimate of the entropy of a random vector," *Problemy Peredachi Informatsii*, vol. 23, no. 2, pp. 9–16, 1987.

[15] L. Paninski, "Estimation of entropy and mutual information," *Neural computation*, vol. 15, no. 6, pp. 1191–1253, 2003.

[16] P. Valiant and G. Valiant, "Estimating the unseen: improved estimators for entropy and other properties," in *Advances in Neural Information Processing Systems*, 2013, pp. 2157–2165.

[17] Y. Wu and P. Yang, "Minimax rates of entropy estimation on large alphabets via best polynomial approximation," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3702–3720, 2016.

[18] P. Jupyter, M. Bussonnier, J. Forde, J. Freeman, B. Granger, T. Head, C. Holdgraf, K. Kelley, G. Nalvarte, A. Osheroff *et al.*, "Binder 2.0-reproducible, interactive, sharable environments for science at scale," in *Proceedings of the 17th Python in Science Conference*, 2018, pp. 113–120.

[19] S. Fujishige, "Polymatroidal dependence structure of a set of random variables," *Information and Control*, vol. 39, no. 1, pp. 55 – 72, 1978.

[20] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, July 1948.

[21] H. Narayanan, *Submodular Functions and Electrical Networks*. Elsevier, 1997, vol. 54.

[22] K. Nagano, Y. Kawahara, and S. Iwata, "Minimum average cost clustering." in *NIPS*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1759–1767.

[23] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2002.

[24] S. Fujishige and S. Isotani, "A submodular function minimization algorithm based on the minimum-norm base," *Pacific Journal of Optimization*, vol. 7, no. 1, pp. 3–17, 2011.

[25] S. Watanabe, "Information theoretical analysis of multivariate correlation," *IBM Journal of Research and Development*, vol. 4, no. 1, pp. 66–82, 1960.

[26] B. A. Galler and M. J. Fisher, "An improved equivalence algorithm," *Communications of the ACM*, vol. 7, no. 5, pp. 301–303, 1964.

[27] R. E. Tarjan and J. Van Leeuwen, "Worst-case analysis of set union algorithms," *Journal of the ACM (JACM)*, vol. 31, no. 2, pp. 245–281, 1984.

[28] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[29] C. Chan, A. Al-Bashabsheh, and Q. Zhou, "Info-clustering: An efficient algorithm by network information flow," *arXiv preprint arXiv:1702.00109*, 2017.

[30] V. Kolmogorov, "A faster algorithm for computing the principal sequence of partitions of a graph," *Algorithmica*, vol. 56, no. 4, pp. 394–412, 2010.

[31] S. Jegelka, H. Lin, and J. A. Bilmes, "On fast approximate submodular minimization," in *Advances in Neural Information Processing Systems*, 2011, pp. 460–468.