Graph Entropy, Posets,

and their applications in

Sorting Under Partial Information

Preetum Nakkiran Matthew Francis-Landau
{preetum, mfl}@berkeley.edu

May 2015

Abstract

In traditional comparison sorting, we are given a set of n elements with no a priori information about their order. Uniquely identifying a permutation requires $\log_2 n! \sim n \log n$ bits, and therefore $\Omega(n \log n)$ comparisons. In the setting of *sorting under partial information (SUPI)*, we are given some initial comparisons, and would like to minimize the number of additional "comparison oracle" calls required to sort, given our partial information. Similar to the traditional setting, we would like to lower-bound the query complexity of algorithms for SUPI, and exhibit algorithms which perform provably close to the lower-bound.

In some sense this is an information-theoretic question: we are given some partial information about the ordering, and want to know how much additional information is required to determine the ordering. The bulk of our survey involves making this precise, which requires the concepts of posets (partially ordered sets) and "graph entropy" (an entropy-like functional on graphs): Our initial comparisons have the structure of a poset, and the lower-bound is related to the graph-entropy of this poset.

In this survey, we review various fundamental properties of posets and graph-entropy, and eventually see their application in proving lower-bounds and optimality of algorithms for SUPI. We try to present all the required ideas from scratch, adding intuition and examples along the way.

1 Introduction

In traditional comparison sorting, we are simply given a set of elements with no a priori information about their order. In the problem of *sorting under partial information*, we are given some initial comparisons P, and we are allowed to make additional oracle calls to a "comparison oracle". We would like to minimize the number of additional oracle calls, given our partial information P.

In the traditional setting, the n elements could be in any of $n!$ permutations. Uniquely identifying the permutation (aka "sorting") requires $\log_2 n! \sim n \log n$ bits, and each comparison yields a single output bit. Therefore we have the obvious lower-bound of $\Omega(n \log n)$ comparisons for sorting under no prior information.

Similarly, for a given set of comparisons P, we can consider the number permutations consistent with P. Let this number of "linear extensions" be $e(P)$. Then we have the analogous lower-bound of $\log_2 e(P)$ additional comparisons required. Notice that for $P = \emptyset$, the number of extensions $e(P) = n!$, and this reduces to the traditional case. In general, we would like to design (preferably efficient) algorithms which provably require close to the lower-bound of $\log_2 e(P)$ additional comparisons.

At a high level, these lower-bounds have a very information-theoretic nature: We start out with high uncertainity about the ordering, and each comparison gives us at most one bit of information about the order.

With partial information P, our initial uncertainty is somehow reduced. Thus we expect the tools of information theory (such as entropy) may be useful here. In traditional sorting, the connection can easily be made precise: Let X be the uniform distribution on permutations, so the entropy $H(X) = \log n!$. Each comparison Y_i is binary-valued, so $H(Y_i) \leq 1$. Let $Y^k = (Y_1, Y_2, \ldots Y_k)$. After k comparisons, the conditional entropy

$$\begin{aligned} H(X|Y^k) &= H(X, Y^k) - H(Y^k) && \text{(by definition)} \\ &\geq H(X, Y) - k && (H(Y_i) \leq 1) \\ &= H(X) - k && (Y^k \text{ is a function of } X) \\ &= \log n! - k \end{aligned}$$

In the end we require $H(X|Y^k) = 0$, so we must have $k \geq \log n!$. Basically, this is a formalization of the intuition "the uncertainty starts at $\log n!$ bits, and can decrease by at most 1 bit in each comparison."

It is not as simple to devise an entropy argument when we have partial information P (we must invent some appropriate distributions, as we did above). The concept of *graph-entropy* is useful here. Roughly, we can think of graph entropy as a coding problem: Given a graph G, we want to encode a uniformly-distributed vertex v. However, unlike classical coding, we only want to distinguish vertices which are adjacent in G. The graph entropy $H(G)$ is the average number of bits required to encode a vertex in G, up to indistinguishability given by the graph. For example, the graph entropy of the complete graph on n vertices is just the classical Shannon entropy of the vertex distribution ($\log n$), since the graph coding problem corresponds exactly to classical coding (all vertices are distinguishable from each other). And the graph entropy of a bipartite graph is 1, since we only need to distinguish which side a vertex belongs to.

We can consider the relations P as having the graph structure of a *poset*, with edges between elements that are comparable in P. We will eventually relate the number of extensions $e(P)$ to the entropy $H(\overline{P})$. Intuitively, a sparse poset P (with few relations) will have many linear extensions, and its complement $\overline{P}$ will be dense (and so have high graph entropy).

Finally, we will show how this information-theoretic interpretation can be used to prove optimality for natural algorithms that sort under partial information, as shown in [1]. For example, the following is a simple "insertion-sort" variant: First find the longest chain in P, then call the comparison oracle to insert all other elements into it. This turns out to require a near optimal number of comparisons. The optimality proof for this is simple, and essential just involves showing that if the algorithm requires many additional comparisons (ie, P has only short chains), then $H(\overline{P})$ is high, so $e(P)$ is high (ie, many comparisons are necessarily required). We also review the simple merge-sort variant given in [1], which is the natural extension of merge-sort to this setting.

Contribution. This survey was written to fully understand the ideas involved in [1], which presents and analyzes algorithms for sorting-under-partial-information (SUPI). The analysis of [1] builds on many years of fundamental results on posets and graph entropy. In this survey, we aim to collapse these results, and present a guided tour of all the required ideas from scratch. We have expanded many of the proofs with elaboration, intuition, and examples. Though we are motivated by applications in SUPI, the development of posets and graph entropy are independently interesting.

2 Poset Polytopes

In a poset P, we have a set of elements $\{x_i\}$ and relationships $x_i < x_j$. Being partially ordered, this means that there exists elements x_i, x_j such that $x_i < x_j$ and $x_j < x_i$ are both consistent with the relationships given. For a given poset P, it is useful to consider the "comparability graph", which has edges between elements if they are comparable in P. This is the transitive closure of P, where each chain becomes a clique. When we say "independent set in P", we mean an independent set in this transitive closure (ie, a set of elements which are pairwise incomparable).

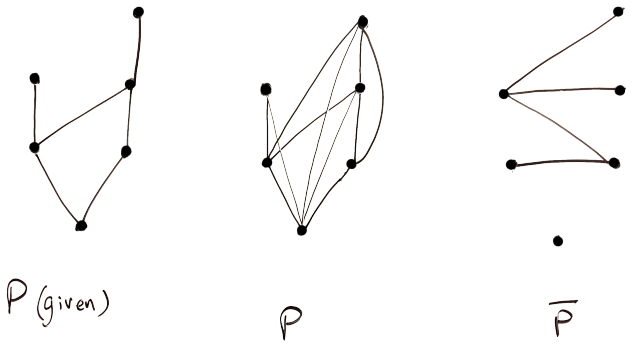


Figure 1: A poset, its transitive closure P, and the complement $\overline{P}$.

The eventual goal of discussing posets is to relate the entropy $H(\overline{P})$ to the number of linear extensions $e(P)$. Intuitively, a sparse poset P (with few relations) will have many linear extensions, and its complement $\overline{P}$ will be dense (and so have high graph entropy). This is made precise through a geometric interpretation of $e(P)$, related to the volume of a certain poset-polytope. Below, we will formally define and examine the following three polytopes:

1. $\mathcal{O}(P)$: The "order polytope", of points whose coordinates respect the poset relations.
2. $VP(P)$: The "vertex-packing polytope", the convex combination of independent sets in P.
3. $\mathcal{C}(P)$: The "chain polytope", of values on the poset such that all chains sum to ≤ 1.

We will investigate the following relations:

1. The volume $|\mathcal{O}(P)| = \frac{e(P)}{n!}$
2. Polytopes $\mathcal{C}(P) = VP(P)$.
3. Volumes $|\mathcal{C}(P)| = |\mathcal{O}(P)|$.

Thus, putting these together, we will have

$$|VP(P)| = |\mathcal{C}(P)| = |\mathcal{O}(P)| = \frac{e(P)}{n!}$$

We will later establish a connection $H(\overline{P}) \longleftrightarrow VP(P)$, thus completing the picture.

2.1 Cast of Characters

Definition 1 *The* order polytope $\mathcal{O}(P)$ *of a poset* P *on* n *elements* $\{p_i\}$ *is:*

$$\mathcal{O}(P) := \{x \in [0,1]^n : p_i < p_j \implies x_i \leq x_j\}$$

That is, vectors in $\mathcal{O}(P)$ are such that their coordinates respect the poset relations of P.

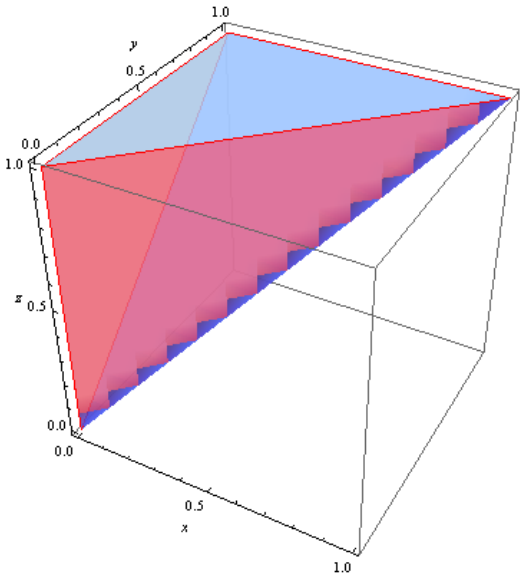


Figure 2: $\mathcal{O}(P)$ for the poset $x \leq y \leq z$ in 3D.

Definition 2 *The* chain polytope $\mathcal{C}(P)$ *of a poset* P *is the convex set:*

$$\mathcal{C}(P) := \{x \in [0,1]^n : \sum_{i \in I} x_i \leq 1 \quad \textit{for all chains } I \textit{ in } P \quad \}$$

We imagine $\mathcal{C}(P)$ as assigning a number $x_i \in [0,1]$ to each point $p_i \in P$, such that the sum along every chain is ≤ 1. Notice this can be thought of as the polytope corresponding to a linear program, so we can reason about vertices/corner-points by our classification of LP solutions.

Definition 3 *The* vertex-packing polytope $VP(G)$ *of a graph* G *is the convex set with vertices* χ_S*, the characteristic vectors of independent sets in* G*.*

$$\begin{aligned} VP(G) &:= conv\{\chi_S : \quad S \textit{ is an independent set in } G \quad \} \\ &= conv\{\chi_S : S \in S(G)\} \end{aligned}$$

Let $S(G)$ be the set of all independent-sets of G. Then $VP(G)$ consists of vectors which can be written $\vec{a} = \sum_{S \in S(G)} \lambda_S \chi_S$ for some λ_S such that $\sum_S \lambda_S = 1$.

2.2 Order and Symmetry

The order polytope $\mathcal{O}(P)$ is a natural geometric realization of a poset P. Here we show:

Lemma 1

$$|\mathcal{O}(P)| = \frac{e(P)}{n!}$$

Proof. This follows from a simple symmetry argument. Pick a point $x \in [0,1]^n$ uniformly at random. Consider the ordering induced by the coordinates of x. (For example, we may have $x_2 < x_1 < x_3 < \dots$). By symmetry, this order is equally likely to be one of $n!$ permutations. Of these permutations, $e(P)$ of them satisfy the relations of P (by definition of $e(P)$). Therefore, the probability $x \in \mathcal{O}(P)$ is $e(P)/n!$. So the volume $|\mathcal{O}(P)|$ is exactly $e(P)/n!$. $\square$

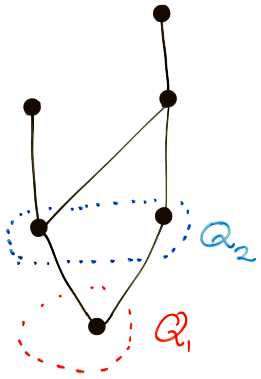


Figure 3: Example of Q_1 and Q_2 in a simple poset.

2.3 Packing Chains

Here we show:

Lemma 2

$$\mathcal{C}(P) = VP(P)$$

Proof. It is sufficient to show that the vertices of $\mathcal{C}(P)$ are exactly the vertices of $VP(P)$ (which by definition are vectors χ_S).

First, clearly the vectors $\chi_S \in \mathcal{C}(P)$, since an independent set S can only contain one vertex from a given chain (so the sum of χ_S along a chain ≤ 1). Further, χ_S is a vertex of $\mathcal{C}(P)$ because each constraint $0 \leq x_i \leq 1$ is tight.

Conversely, we will show that an $x \in \mathcal{C}(P)$ that is not a vertex of $VP(P)$ is also not a vertex in $\mathcal{C}(P)$. We have $x \neq \chi_S$ for all independent sets S, since x is not a vertex of $VP(P)$. We will show how to write x as a convex combinations of "perturbed" vectors $x_\pm \in \mathcal{C}(P)$. First, if all $x_i \in \{0, 1\}$, then the 1-indices of x correspond to an independent set (since the constraint $\sum_{i \in I} x_i \leq 1$ forces at most 1 vertex from each chain $\longleftrightarrow$ clique). Hence, the set $Q = \{p_i \in P : x_i \in (0, 1)\}$ is non-empty, and forms a subposet of P. (The notational convention is x_i is the value assigned to point p_i in the poset). Let Q_1 be the minimal elements of poset Q, and let Q_2 be minimal elements of subposet $Q \setminus Q_1$. Notice Q_1 is nonempty by definition. See Figure 3.

First we consider the trivial case, let Q_2 be empty, we can then easily show that x is not a vertex of $\mathcal{C}(P)$: $Q_2 = \emptyset \implies Q_1 = Q$, and all the elements of Q are minimal (w.r.t. the poset order P). This means there are no relations in Q. Therefore, we can shift all the $\{x_i : p_i \in Q_1 \cup Q_2\}$ by $\pm\varepsilon$ while staying in $\mathcal{C}(P)$ (chains in P are just an element of Q, and possibly some additional element with value $x_i = 0$, not affecting the sum).

Now we consider the case where Q_2 is nonempty. We will perturb points in Q_1 by $\pm\varepsilon$ and point in Q_2 by $\mp\varepsilon$. Notice that every affected chain in P contains exactly one element of Q_1, and one element of Q_2.

More formally, let $\varepsilon = \min\{x_i, 1 - x_i : x_i \in Q_1 \cup Q_2\}$. We can then construct a new x_is such that

$$x_i^{(1)} = \begin{cases} x_i + \varepsilon & x_i \in Q_1 \\ x_i - \varepsilon & x_i \in Q_2 \\ x_i & \text{otherwise} \end{cases}$$

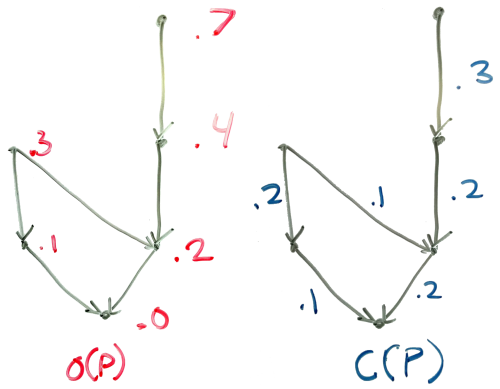


Figure 4: Comparision for simple ordering between $\mathcal{O}(P)$ and $\mathcal{C}(P)$

$$x_i^{(2)} = \begin{cases} x_i - \varepsilon & x_i \in Q_1 \\ x_i + \varepsilon & x_i \in Q_2 \\ x_i & \text{otherwise} \end{cases}$$

We can see that these new x_is will preserve our sum constraints since each chain of length 2 or more will have one of its vertices in each Q_1 and Q_2, therefore the sum for these constrains will still be less then 1. For p_i with no relations, they must already have a $x_i + \varepsilon \leq 1$ by the definition of ε. We can then see that $x_i = \frac{1}{2}(x_i^{(1)} + x_i^{(2)})$ which means that p_i must not be a vertex. □

2.4 Chains in Order

We will now establish a connection between $\mathcal{C}(P)$ and $\mathcal{O}(P)$ by creating a bijection between these two sets (as in [2]). To go from $\mathcal{O}(P)$ to $\mathcal{C}(P)$ we can define ϕ such that it takes points $x \in \mathcal{O}(P) \subseteq [0,1]^n$ (where components respect the poset order), and converts these to points in $\mathcal{C}(P)$ (where all chains sum to ≤ 1). Intuitively, ϕ will act as a "gradient" operator, computing differences between adjacent elements. Then the inverse ϕ^{-1} will act as an "integral", summing the differences along a chain to recover the original values. See Figure 4.

Formally, we can first observe that for a given chain $0 \leq x_{i_0} < x_{i_2} < ... < x_{i_n} \leq 1$, therefore if we were to sum the differences between these elements, we must have $\sum_j (x_{i_j} - x_{i_{j-1}}) = x_{i_n} \leq 1$. Hence we can define ϕ as:

$$\phi(x_i) = \min\{x_i - x_j : p_i > p_j\}$$

Notice ϕ will send $\mathcal{O}(P) \to \mathcal{C}(P)$, since for an assignment of values $x \in \mathcal{O}(P)$, summing the "differences" $\phi(x)$ along a chain to point p_i will recover the original value x_i, which will be ≤ 1 since all $x \in [0,1]^n$. Thus the sum along chains of $\phi(x)$ is always ≤ 1, and $\phi(\mathcal{O}(P)) \subseteq \mathcal{C}(P)$. We can see that this function is continuous on all of $\mathcal{O}(P)$ and that it is linear when not crossing a "boundary" representing $p_i < p_j$ (so, it is piecewise linear).

For ϕ^{-1} we can see that summing along the "largest valued path" up to point p_i will recover x_i. This follows from expression above: $\sum_j (x_{i_j} - x_{i_{j-1}}) = x_{i_n}$, and the fact that we are selecting the minimum

$$\phi^{-1}(\phi(x_i)) = \max_j \{\phi(x_i) + \phi(x_{j_1}) + \cdots + \phi(x_{j_n}) : p_i > p_{j_1} > ... > p_{j_n}\}$$

This function ϕ is a bijection, thus establishing a connection between these two sets.

To further extend this connection to show that $|\mathcal{C}(P)| = |\mathcal{O}(P)|$, we need some rather involved algebraic arguments (involving polynomials that count integer lattice-points contained in these polytopes). These

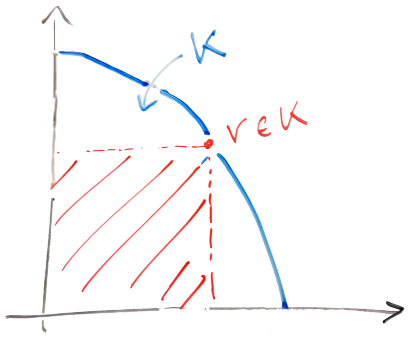


Figure 5: Example of a convex corner in 2D.

proofs are given in [2], however they do not provide much intuition, and use somewhat obscure techniques, so we do not review them here.

Conclusion. Given the results in the above sections, we see that:

$$|VP(P)| = |\mathcal{C}(P)| = |\mathcal{O}(P)| = \frac{e(P)}{n!}$$

2.5 Cornered

From the above discussion, we have estabilished (modulo an ommitted algebraic argument) that

$$|VP(P)| = \frac{e(P)}{n!} \tag{1}$$

Later, we will want to relate $H(\overline{P})$ to the volume $|VP(P)|$. To do this, we will need some bounds on $|VP(P)|$. The following notion will be useful:

Definition 4 *A* convex corner K *is a convex set* $K \subseteq \mathbb{R}^n_+$ *such that for all* $v \in K$*:*

$$\vec{0} \leq v' \leq v \implies v' \in K$$

Where the ordering $\leq$ *is element-wise.*

That is, if $v \in K$, then the entire cube with corners $\vec{0}, v$ lies in K. For example, see Figure 5.

This is useful because:

Lemma 3 *If* K *is a convex corner, and* $v \in K$*, then the volume*

$$|K| \geq \prod_i v_i$$

Proof. This is easy to see (eg, in Figure 5), since K contains the cube from 0 to v, with volume $\prod_i v_i$. □

We now show:

Lemma 4 $VP(P)$ *is a convex corner.*

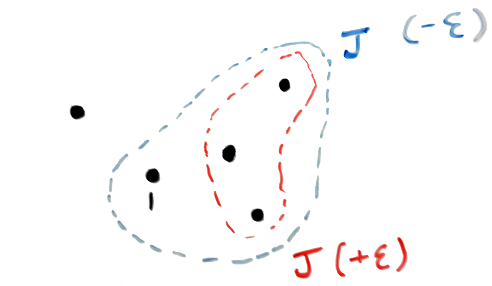


Figure 6: Illustration of convex corner proof. Edges not shown.

Proof. We must show that given some $v \in VP(P)$, all coordinate-wise lower $v' \leq v$ are also on $VP(P)$. Recall $VP(P)$ is a convex combination of independent sets, so:

$$v = \sum_{S \in S(G)} \lambda_S \chi_S \quad \text{where} \quad \sum_S \lambda_S = 1$$

Consider decreasing v by some small ε in only the first coordinate $v_1 \mapsto v_1' = v_1 - \varepsilon$. We will show how to write v' as a convex combination $v' = \sum_{S \in S(G)} \lambda'_S \chi_S$, thus showing $v' \in VP(P)$. Pick some independent set J containing the point 1, for which $\lambda_J > 0$. Let $\widetilde{J} = J \setminus \{1\}$, and notice $\widetilde{J}$ is also an independent set (as a subset of J). We will decrease the first coordinate by decreasing the coefficient λ_J (which decreases the values of all points in J), then compensating by increasing the coefficient of $\lambda_{\widetilde{J}}$ (to leave other points unaffected). For example, see Figure 6.

Mathematically, define:

$$\lambda'_\bullet : \begin{cases} \lambda'_J = \lambda_J - \varepsilon \\ \lambda'_{\widetilde{J}} = \lambda'_{\widetilde{J}} + \varepsilon \\ \lambda'_S = \lambda_S \quad \text{(otherwise)} \end{cases}$$

Notice that this achieves the intended affect: only the first coordinate of $v' = \sum_{S \in S(G)} \lambda'_S \chi_S$ is changed (by $-\varepsilon$). Further, the sum is identical: $\sum_S \lambda'_S = \sum_S \lambda_S = 1$. Therefore $v' \in VP(P)$. This procedure can be repeated for other coordinates (and ε can be chosen small enough), so all coordinate-wise lower $v' \leq v$ are in $VP(P)$, and $VP(P)$ is a convex corner. □

Finally, we will need the following simple geometric claim:

Lemma 5 *For any $a \in \mathbb{R}^n, a_i \geq 0$, the volume of the "corner":*

$$L := \{x \in \mathbb{R}^n : x_i \geq 0, \quad x \cdot a \leq 1\}$$

is

$$|L| = \frac{1}{n!} \prod_i \frac{1}{a_i}$$

Notice that this corner intersects the i-th axis at $\frac{1}{a_i}$ (For example, imagine in 3-dimentions).

Proof. This is easy to show by simple calculus. (It is sufficient to show that the "unit corner"

$$L_0 := \{x \in \mathbb{R}^n : x_i \geq 0, \quad \sum_i x_i \leq 1\}$$

has volume $\frac{1}{n!}$. Then the general case follows by scaling argument.) $\square$

3 Entropy and Information Theory

Here we briefly review some aspects of classical information theory. This will be fundamental in our eventual application of sorting under partial information, since we want to quantify how much "information" a given poset P tells us about the actual ordering.

The reader already familiar with basic entropy and mutual-information can skip this section.

3.1 Shannon Entropy

The field of Information Theory started with Shannon's 1948 paper *A Mathematical Theory of Communication* [3]. Intuitively, the key idea is that the "information content" of a message is determined by its "uncertainity", in the sense that a message with high uncertainty (say, one that could take many possible values uniformly) reveals a lot of information when it is received.

Thus random variables enter the picture, and we would like to quantify the information-content of a discrete r.v. X by some functional $H(X)$ (the "entropy"). Clearly this functional should not depend on the domain of X, but only its distribution (messages taking values ± 1 should not be fundamentally different than messages taking 0/1). Let $p_1, p_2, \ldots, p_n$ be distribution of X (eg, $p_i = \Pr[X = i]$). Then we want to define an appropriate real-valued function $H(p_1, p_2, \ldots, p_n)$. What properties would we like from H?

First, let us consider only uniform distributions. Let $H(U_n)$ denote $H(1/n, \ldots, 1/n)$, ie the entropy of a uniform distribution on n-elements. For example, $H(U_2) = H(1/2, 1/2)$ is the entropy of a single uniformly random bit. A "logarithmic" measure of information is natural in this setting: We would like the information of two **independent** uniform bits to be twice the information of a single bit. This joint distribution is U_4, so we want:

$$H(U_4) = 2H(U_2)$$

And in general, taking N independent copies of an information source should give us N times the information. For the uniform binary case, this is:

$$H(U_{2^N}) = NH(U_2)$$

We can continue investigating properties we want in this way. Shannon specified the following 3 desirable properties of H (reproduced verbatim from [3]):

1. H should be continuous in the p_i.

2. If all the p_i are equal, $p_i = \frac{1}{n}$, then H should be a monotonic increasing function of n. With equally likely events there is more choice, or uncertainty, when there are more possible events.

3. If a choice be broken down into two successive choices, the original H should be the weighted sum of the individual values of H. The meaning of this is illustrated in Fig. 6. At the left we have three

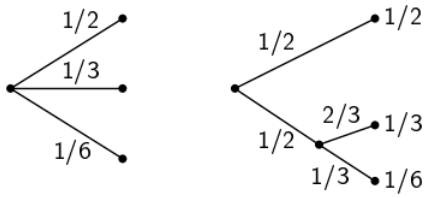


Fig. 6—Decomposition of a choice from three possibilities.

possibilities $p_1 = \frac{1}{2}, p_2 = \frac{1}{3}, p_3 = \frac{1}{6}$. On the right we first choose between two possibilities each with probability $\frac{1}{2}$, and if the second occurs make another choice with probabilities $\frac{2}{3}, \frac{1}{3}$. The final results have the same probabilities as before. We require, in this special case, that

$$H(\tfrac{1}{2}, \tfrac{1}{3}, \tfrac{1}{6}) = H(\tfrac{1}{2}, \tfrac{1}{2}) + \tfrac{1}{2} H(\tfrac{2}{3}, \tfrac{1}{3}).$$

The coefficient $\frac{1}{2}$ is the weighting factor introduced because this second choice only occurs half the time.

Most of the content is in the third property, which generalizes our discussion of the iid uniform case to distributions which can be "factored." In fact, these 3 properties are sufficient to fully characterize the entropy function H (up to a factor):

Theorem 1 *(Shannon [3]) The only functional H satisfying the above three properties is*

$$H = K \sum_i p_i \log\left(\frac{1}{p_i}\right)$$

For any positive constant K.

For a r.v. X with support size n, the entropy $H(X)$ is maximized when X is uniformly distributed (then $H(X) = \log n$) – this is easy to see from some calculus.

3.2 Related measures

The notion of *conditional entropy* $H(X|Y)$ will also be useful, as intuitively the uncertainty of X if we know Y. It is just the expected entropy of the conditional distribution $P(X|Y)$, with expectation over Y:

$$H(X|Y) := \mathbb{E}_Y[H(X|Y = y)]$$

Conditioning never increases entropy: $H(X|Y) \leq H(X)$.

The entropy of a joint distribution (X, Y) satisfies the identities

$$H((X, Y)) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

Intuitively, the uncertainty of (X, Y) is the uncertainty of X, plus the additional uncertainty of Y after knowing X. For example, if Y is a function of X, then $H(Y|X) = 0$, so $H((X, Y)) = H(X)$. Combining the joint-distribution identity with the fact that conditioning does not increase entropy, we have the following *sub-additivity*:

$$H((X, Y)) \leq H(X) + H(Y)$$

Further, if X, Y are independent, then $H(Y|X) = H(Y)$, and we recover $H((X, Y)) = H(X) + H(Y)$ as we may expect.

The concept of *mutual information* captures the "shared information" or "shared randomness" between two random variables. The mutual information $I(X; Y)$ between two r.v.s X, Y can be defined (symmetrically) as

$$I(X; Y) = H(X) + H(Y) - H((X, Y)) = H(X) - H(X|Y)$$

Intuitively, this is the uncertainty of X, minus the uncertainty that remains after given Y. For example, if X, Y are independent, then $H(X|Y) = H(X)$, so $I(X; Y) = 0$. If X is a function of Y, then $H(X|Y) = 0$, so $I(X; Y) = H(X)$ (all the randomness of X is shared with Y).

3.3 Source Coding

One of the most useful (and intuitive) applications of entropy is due to its **operational characterization**, as roughly the number of bits required to encode a message. This is formalized as the **Source Coding Theorem**, which implies: *To encode N iid instances of a discrete random variable X, it is necessary and sufficient to use $NH(X)$ bits.* Formally, this is an asymptotic result (as $N \to \infty$), and we must allow some negligible probability of decoding error. Intuitively, we think of $H(X)$ as the number of bits required (in the long run) to encode the r.v. X. We call the support of X the set of "symbols" which we wish to encode.

Many properties of entropy have coding-theoretic interpretations. For example, the identity

$$H(X, Y) = H(X) + H(Y|X)$$

translates as "number of bits required to encode (X, Y) is the bits required to encode X, plus the additional bits to encode Y, if given X". [1]

Huffman codes are a particularly simple construction, which minimize the expected codeword length among prefix-free codes. It is easy to show (via constrained minimization) that Huffman codes achive an expected codeword length of $\leq H(X) + 1$. (The $+1$ gap is due to non-asympototic effects).

4 Graph Entropy

4.1 Intuition

It is easiest to understand Graph Entropy by extending the operational definition of classical entropy (via the source-coding theorem). In the classical case, we treated all symbols as distinct, and required an encoding that distinguished among them (up to negligible error). Instead, let us consider a "distinguishability graph" G, with vertices as symbols, and edges between symbols that we are required to distinguish. (Note, no further structure is assumed – "indistinguishability" need not be transitive.). Let P be a probability distribution on vertices (symbols). Consider a source that emits iid symbols according to distribution P. Then the graph entropy $H(G, P)$ is the number of bits necessary and sufficient to encode these symbols, up

[1] Making such coding statements formal involves adding a generous helping of "on average" and "asymptotically", etc.

to indistinguishability given by G. That is, if we pick a random v vertex according to P, we need $H(G, P)$ bits to encode v, assuming we only need to distinguish it from other adjacent vertices in G.

We do not state this this coding-theoretic definition formally, since we will not need it in what follows, but it is a good picture to keep in mind.

4.2 Formal Definitions

Here we give two equivalent definitions of the graph entropy $H(G, P)$. The notation $H(G)$ refers to $H(G, P)$ for P uniform on vertices: $H(G) = H(G, U_n)$.

$$H(G, P) := \min_{\substack{X,Y: X \sim P, \\ X \in Y \in S(G)}} I(X; Y) \tag{2}$$

$$H(G, P) := \min_{a \in VP(G)} \sum_i p_i \log\left(\frac{1}{a_i}\right) \tag{3}$$

In equation (2) we consider joint distributions (X, Y) where X is a vertex (with marginal probability P) and Y an independent set containing X. The entropy $H(G, P)$ is the minimum mutual information $I(X; Y)$, among all such distributions. (Recall, mutual information roughly represents the amount of information that we can gather about one variable, say X, by knowing the other, Y.) From the coding-characterization, we expect that independent sets appear in this definition, since they represent "mutually indistinguishable symbols" in G. For example, if the mutual information $I(X; Y)$ is always high, then every independent set Y in G reveals a lot of information about which random vertex $X \in Y$ was chosen (ie, all independent sets are small, and the distinguishability graph is dense). More explicit examples are in the following section.

In equation (3) we consider minimizing a "entropy-like" quantity over points a in the "vertex-packing" polytope $VP(G)$ (convex combinations of independent sets). Roughly, the particular convex combination defines a distribution on independent sets, similar to Y in the previous definition. This equivalence will be made formal in a later section.

4.3 Examples

Here we will cover some simple examples to build intuition for the formal definitions. Parts of these examples were adapted from [4].

Example 1 *In an empty graph, $H(G) = 0$.*

The intuition is that there are no symbols that we need to distinguish between, and as such we do not need to send any information to identify a symbol.

Proof. Consider a particular joint distribution where X is uniform on n vertices (as required), and Y is always the entire vertex set. This gives us an upper-bound $H(G) \leq I(X; Y) = H(X) - H(X|Y) = \log n - \log n = 0$. Further, mutual information is always non-negative, so $H(G) = \min_{\ldots} I(X; Y) \geq 0$. □

Let us now consider the minimizer $a \in VP(G)$ in the polytope definition of $H(G)$. Since the entire graph is an independent set, the all-one vector $1 \in VP(G)$. This clearly minimizes the quantity in equation (3), since each term in the sum is 0.

Example 2 *For a clique $H(K_n) = \log n$.*

Intuitively, if we have a graph that is fully connected and all symbols occur with equal probability, then there is no way to "compress" a message, and thus each message will require $\log_2 n$ bits.

Proof. For the upper-bound, consider the particular joint distribution where X is a uniform vertex, and Y is the independent set containing only X. This gives

$$\begin{aligned} H(G) \leq I(X;Y) &= H(X) - H(X|Y) \\ &= \log n - H(X|Y) && (X \text{ uniform}) \\ &= \log n - 0 && (X \text{ is determined by } Y) \end{aligned}$$

For the lower-bound, notice that for any valid joint distribution (X, Y), the independent set Y can only be the set containing just X. Therefore $H(X|Y) = 0$ always, and $I(X, Y) = H(X) - H(X|Y) = H(X) = \log n$ for all joint distributions (X, Y).□

Now consider the minimizer $a \in VP(G)$. The only independent sets in G are individual vertices, so points $a \in VP(G)$ are exactly distributions on vertices. By symmetry considerations, we can see [2] that the minimizer is the uniform distribution $a_i = 1/n$.

Example 3 *For complete balanced bipartite graphs, $H(K_{n,n}) = 1$.*

Intuitively, we only need to distinguish if a vertex is on the left side (A) or right side (B) of the graph (requiring 1 bit).

Proof. For the upper-bound, consider the particular joint distribution where X is a uniform vertex (on $2n$ verts), and Y is the largest independent set containing X (ie, A or B). We have

$$\begin{aligned} H(G) \leq I(X;Y) &= H(X) - H(X|Y) \\ &= \log 2n - H(X|Y) && (X \text{ uniform}) \\ &= \log 2n - \log n && (\star) \\ &= 1 \end{aligned}$$

Step $\star$ is because the conditional distribution of X given its containing set Y ($= A$ or B) is uniform over Y, which has size n.

For the upper-bound, notice that every the largest independent set in $K_{n,n}$ is of size at most n. So for any valid (X, Y), the conditional distribution of X given YF has support on at most n vertices. Therefore the entropy $H(X|Y)$ is at most $\log n$, and we have $I(X;Y) = H(X) - H(X|Y) \geq \log 2n - \log n = 1$. □

Let us again consider the corresponding minimizer $a \in VP(G)$. It turns out to be a combination of the two maximal independent set vectors: $a = \frac{1}{2}\chi_A + \frac{1}{2}\chi_B$. This gives $a_i = \frac{1}{2}$. For uniform $p_i = \frac{1}{2n}$, we have

$$\begin{aligned} H(G, P) &= \sum_i p_i \log\left(\frac{1}{a_i}\right) \\ &= \frac{1}{2n} \sum_{i \in [2n]} \log(2) \\ &= 1 \end{aligned}$$

In Section 4.5, we will see how to systematically map between these two definitions, as we have been doing for these specific cases.

We now easily show:

[2] Or at least, we intuitively believe...

Lemma 6 *If all independent sets in G have size $\leq \alpha$, then the entropy $H(G) \geq \log\left(\frac{n}{\alpha}\right)$.*

Proof. For any valid joint distribution (X, Y), the conditional distribution $P(X|Y)$ has support on at most α vertices (for all Y), since all independent sets Y have $|Y| \leq \alpha$. Therefore the entropy of the conditional distribution $H(X|Y = y) \leq \log \alpha \quad \forall y$, and thus the conditional entropy

$$H(X|Y) = \mathbb{E}_y[H(X|Y = y)] \leq \log \alpha$$

Then we have

$$I(X;Y) = H(X) - H(X|Y) \geq \log n - \log \alpha \quad \forall (X, Y)$$

□

Again, this should be intuitive: if G has no large independent sets, then most its vertices are distinguishable, so the entropy must be high.

As a direct corollary, we have:

Lemma 7 *(From Lemma 5 in [1]). For a maximal chain C in poset P, we have*

$$|C| \geq n2^{-H(\overline{P})}$$

Proof. If C is maximal, the largest clique in P is of size $|C|$, so the largest independent set in $\overline{P}$ is of size $|C|$. Then by Lemma 6, we have

$$H(\overline{P}) \geq \log\left(\frac{n}{|C|}\right)$$

Rearranging this, we have the result. □

One final example (which will be useful later):

Example 4 *For the complement of $K_{n,n}$, that is two cliques $G = K_n \sqcup K_n$, we have $H(G) = \log n$, and the minimizer $a_i = \frac{1}{n}$.*

Proof. Let $G = A \sqcup B$. For the upper-bound, consider the joint distribution defined by picking vertex X uniformly, then picking Y by including X and a random vertex in the other clique. The conditional distribution $P(X|Y)$ is supported uniformly on the two vertices in Y, so $H(X|Y) = \log 2 = 1$. Then

$$H(G) \leq I(X;Y) = H(X) - H(X|Y) = \log(2n) - 1 = \log n$$

For the lower-bound, the largest independent set in G is of size 2, so we conclude by Lemma 6.

We defer the formal proof of the minimizer here (it will follow from the general mapping in Section 4.5). But notice that a is a uniform combination of all the independent set vectors:

$$\vec{a} = \frac{1}{|S(G)|} \sum_{S \in S(G)} \chi_S = \frac{1}{n^2} \sum_{a \in A, b \in B} \chi_{a,b} = [1/n, \cdots, 1/n]$$

And notice that in construction of (X, Y) above, the marginal distribution of Y was uniform over all independent sets. □

4.4 Aside: Other Properties and Applications

Graph entropy satisfies many properties similar to classical entropy (as may be expected from the coding characterization). For example, for two graphs G_1, G_2 on the same vertex set, graph-entropy is *sub-additive* and *monotonic*:

$$H(G_1) \leq H(G_1 \cup G_2) \leq H(G_1) + H(G_2)$$

(Both are easy to understand from the coding perspective). One neat application from lecture notes [5] and [6] (unrelated to sorting) shows:

Claim: We need at least $\log n$ bipartite graphs to cover the complete graph K_n. That is, we want to find bipartite graphs $\{G_i\}$ such that $K_n = G_1 \cup G_2 \cup \cdots \cup G_k$.

Proof. We have seen that $H(K_n) = \log n$, and for any bipartite graph, $H(G_i) \leq 1$. Therefore, by sub-additivity, $\log n = H(K_n) = H(G_1 \cup \cdots \cup G_k) \leq k$. So we must have $k \geq \log n$. □

4.5 Equivalence Proof

We will now show that definitions (2) and (3) for graph entropy are in fact equal. At a high level, we will take a pair (X, Y) in (2), and map it to a point $a \in VP(G)$ in (3), of not greater objective value. Conversely, we will take a point $a \in VP(G)$, and convert it into a valid joint distribution (X, Y) with not greater objective value in (2).

4.5.1 Mappings

The two mappings will be:

1. $(X, Y) \to VP(G)$: Given some joint distribution (X, Y), let $\lambda_S = \Pr[Y = S]$, the marginal distribution of the independent set Y. Then simply let λ_S be the weights of the convex combination: $a = \sum_S \lambda_S \chi_S \in VP(G)$.
2. $VP(G) \to (X, Y)$: Given $a = \sum_S \lambda_S \chi_S$, define the joint distribution (X, Y) as follows: First, pick a random vertex $X \sim P$. Then, pick a random independent set Y among independent sets S which contain X, with probabilities proportional to λ_S.

It is instructive to see how these mappings manifest in the examples we gave from Section 4.3. We will use these two mappings in the formal argument below (showing that neither map increases the objective value).

4.5.2 Reduction

We now formally show that the mappings defined above are non-increasing w.r.t. the two objectives. (Expanding on the proof in [7]). Let define $S(G)$ be the set of independent sets of G. We will first examine the expressions $\min_{X \in Y \in S(G)} I(X; Y) \geq \min_{a \in VP(G)} - \sum_{i=1}^{n} P(X = i) \log a_i$. Suppose that the pair (X, Y) represents the minimal value of the left hand side.

$$
\begin{aligned}
I(X;Y) &= H(X) - H(X|Y) && \text{(By definition of mutual information)} \\
&= -\sum_i P_X(i) \log P_X(i) - \sum_j P_Y(j) H(X|Y=j) \\
&= -\sum_i P_X(i) \log P_X(i) + \sum_j P_Y(j) \sum_i P_{X|Y}(i|j) \log P_{X|Y}(i|j) \\
&= \sum_i P_X(i) \left[-\log P_X(i) + \sum_j P_{Y|X}(j|i) \log P_{X|Y}(i|j) \right] \\
&= \sum_i P_X(i) \left[\sum_j P_{Y|X}(j|i) \log \frac{P_{Y|X}(j|i) P_X(i)}{P_Y(j) P_X(i)} \right] \\
&= -\sum_i P_X(i) \left[\sum_j P_{Y|X}(j|i) \log \frac{P_Y(j)}{P_{Y|X}(j|i)} \right] \\
&\geq -\sum_i P_X(i) \log \sum_{i \in j \in S(G)} P_Y(j) && \text{(Using the fact of that log is concave)}
\end{aligned}
$$

For the reverse direction, suppose that we have the vector $a \in VP(G)$ such that it minimizes the expression $-\sum_i P(X = i) \log a_i$. Additionally, we can use a_i to define Y' such that $\sum_{i \in J \in S(G)} P_{Y'}(j) = a_i$, and $P_{Y'|X}(j|i) = P_{Y'}(j)/a_i$. We also will define $P_{Y^*}(j) = \sum_i P_X(i) P_{Y'|X}(j|i)/a_i$.

$$
\min_{Y \in S(G), X \in Y} I(X;Y) \leq -\sum_i P_X(i) \sum_{j \in S(G), i \in j} \log \frac{P_{Y^*}(j)}{P_{Y'|X}(j|i)}
$$

Using the fact that log is concave, we can see that:

$$
\sum_{j \in S(G)} P_{Y^*}(j) \log \frac{P_{Y'}(j)}{P_{Y^*}(j)} = \sum_{j \in S(G)} \left(\sum_i P_{Y'|X}(j|i) P_X(i) \right) \log \frac{P_{Y'}(j)}{\sum_i P_{Y|X}(j|i) P_X(i)} \leq 0
$$

Which then allows us to write:

$$
-\sum_{i,j} P_X(i) P_{Y'|X}(j|i) \log P_{Y^*}(j) \leq -\sum_{i,j} P_X(i) P_{Y'|X}(j|i) \log P_{Y'}(j)
$$

$$
\min_{X \in Y \in S(G)} I(X;Y) \leq -\sum_{i,j} P_X(i) P_{Y'|X}(j|i) \log \frac{P_{Y'}(j)}{P_{Y'|X}(j|i)} = \sum_i P_X(i) \log a_i
$$

This show both directions of the inequality, and thus proves $\min_{X \in Y \in S(G)} I(X;Y) = \min_{a \in VP(G)} -\sum P_X(i) \log a_i$.

□

4.6 Complementary Entropy

Here we state (mostly without proof) some results relating the entropy of a graph (or poset) to its complement. These will be used in later sections, and are intuitively believable.

For perfect graphs G, *it turns out* that

$$H(G,p) + H(\overline{G},p) = H(p) \tag{4}$$

(A rather involved proof is in [8]). *It turns out* that posets are perfect, so we have

$$H(P) + H(\overline{P}) = \log n \tag{5}$$

We now consider the structure of the minimizer in the definition of graph entropy.

Definition 5 *Let $a(P)$ denote the point $a \in VP(G)$ which achieves the minimum in definition (2) of graph entropy.*

We will need the following complementary result:

Lemma 8

$$a(P)_i a(\overline{P})_i = \frac{1}{n} \quad \forall i$$

(According to [9], this is proven in [8]). We could not track down nor figure out this proof, so instead we will do:
Proof (by two examples).

1. Consider $P = K_n$. In Examples 2 and 1, from Section 4.3, we saw that the minimizers $a(P)_i = \frac{1}{n}$ and $a(\overline{P})_i = 1$. So $a(P)_i a(\overline{P})_i = \frac{1}{n}$.
2. Consider $P = K_{n,n}$. In Examples 3 and 4 from Section 4.3, we saw that the minimizers $a(P)_i = \frac{1}{2}$ and $a(\overline{P})_i = \frac{1}{n}$. So $a(P)_i a(\overline{P})_i = \frac{1}{2n}$.

□

We can show the following related[3] result:

Lemma 9

$$\sum_i a_i b_i \leq 1 \quad \forall a \in VP(G), b \in VP(\overline{G})$$

We may expect something like this to be true, since there is a tension between independent sets in a graph and its complement.

Proof. Consider any two independent sets $A \in G$ and $B \in \overline{G}$, with corresponding vectors χ_A, χ_B. Notice that there can be at most one vertex in the intersection $A \cap B$, since all vertices in the intersection must be both a clique and an independent set in G. Therefore $\chi_A \cdot \chi_B \leq 1$ for all such A, B.

So we have $\sum_i a_i b_i \leq 1$ holds for all vertices a of $VP(G)$ and vertices b of $VP(\overline{G})$. And thus it holds for convex combinations of these vertices: Let $a = \sum_{A \in S(G)} \lambda_A \chi_A$ and $b = \sum_{B \in S(\overline{G})} \lambda'_B \chi_B$. Then

$$\begin{aligned} a \cdot b &= \sum_{A \in S(G), B \in S(\overline{G})} \lambda_A \lambda'_B \chi_A \cdot \chi_B \\ &\leq \sum_{A \in S(G), B \in S(\overline{G})} \lambda_A \lambda'_B \\ &= 1 \qquad (\lambda. \text{ are a distribution}) \end{aligned}$$

□

[3] Related according to [9], at least.

5 Graph Entropy and the Counting Bound

Here we finally establish the relation between $H(P)$ and $e(P)$:

$$\log e(P) = \Theta(nH(\overline{P}))$$

More precisely, we will prove the following strong theorem:

Theorem 2 *(From Theorem 1.1 from [9])*

$$nH(\overline{P}) \geq \log e(P) \geq CnH(\overline{P})$$

for $C \approx 0.09$.

Recall from Section 4.6 that $H(\overline{P}) = \log n - H(P)$. The constant C appears because it is easier to consider $\log(n!) \geq Cn \log n$. Therefore, Theorem 2 follows if we can show:

$$n(\log n - H(P)) \geq \log e(P) \geq \log(n!) - nH(P) \geq Cn(\log n - H(P))$$

which was the original form in [9]. From our discussion on poset polytopes, we know $e(P) = n!\ |VP(P)|$. Therefore, we can equivalently prove

$$\begin{array}{lll} n(\log n - H(P)) & \geq \log e(P) & \geq \log(n!) - nH(P) \\ n^n\ 2^{-nH(P))} & \geq e(P) & \geq n!\ 2^{-nH(P)} \\ \frac{n^n}{n!}\ 2^{-nH(P))} & \geq |VP(P)| & \geq 2^{-nH(P)} \end{array}$$

Proof. (From Section 3 of [9]) To prove Theorem (2), we prove that

$$\frac{n^n}{n!}\ 2^{-nH(P))} \geq |VP(P)| \geq 2^{-nH(P)}$$

For the lower-bound, let $a = a(P)$ (that is, the minimizer $a \in VP(G)$ of the graph entropy objective (3)). Since $VP(G)$ is a convex corner and $a \in VP(G)$, we have the volume lower-bound from Lemma 3:

$$|VP(G)| \geq \prod_i a_i$$

Since a is the minimizer of (3), we also have

$$\begin{aligned} H(P) &= \sum_i \frac{1}{n} \log\left(\frac{1}{a_i}\right) \\ &= -\frac{1}{n} \log\left(\prod_i a_i\right) \\ 2^{-nH(P)} &= \prod_i a_i \end{aligned} \tag{6}$$

Combining these, we have one side of the result:

$$|VP(G)| \geq \prod_i a_i = 2^{-nH(P)}$$

For the upper-bound: Let minimizer $b = a(\overline{P})$. By Lemma 9, every $a \in VP(G)$ satisfies $\sum_i a_i b_i \leq 1$. Therefore, the set L (below) contiains $VP(G)$:

$$VP(G) \subseteq L := \{a \in \mathbb{R}^n : \sum_i a_i b_i \leq 1\}$$

L is a "simple corner" as in Lemma 5, so we can find its volume in terms of $b_i = a(\overline{P})_i$, then connect this to $a(P)$ via the "complementary" Lemma 8, then finally relate to $H(P)$ by Equation 6:

$$\begin{aligned}
|VP(G)| &\leq |L| \\
&= \frac{1}{n!} \prod_i \frac{1}{b_i} && \text{(Lemma 5)} \\
&= \frac{1}{n!} \prod_i n a(P)_i && \text{(Lemma 8)} \\
&= \frac{n^n}{n!} \prod_i a(P)_i \\
&= \frac{n^n}{n!} 2^{-nH(P)} && \text{(Equation (6))}
\end{aligned}$$

Thus showing the upper-bound. □

Now we have shown the fundamental relation $\log e(P) \approx nH(\overline{P})$, [4] connecting number of linear extensions to graph entropy. This will be very useful, since it will be easier to reason about bounds on entropy $H(\overline{P})$ via simple graph-theoretic arguments (about independent sets in $\overline{P}$, or cliques in P), which we can then translate into bounds on $e(P)$.

6 Insertion sort

We finally have the tools required to analyze the query-complexity of algorithms for sorting-under-partial-information (SUPI).

Here we present and analyze the first simple sorting algorithm presented in [1], which will turn out to require an almost-optimal number of additional queries. The algorithm itself is very natural, and the analysis is an easy corollary of the above (powerful) results.

6.1 Algorithm

We first find the longest chain C in the given poset P (ie, longest sequence of given comparisons $x_i \leq x_j \leq x_k \leq \cdots \leq x_m$). [5] Then we insert all other elements into C by making additional comparison-queries.

Algorithm 1 (From [1]). Insertion sort-like algorithm for sorting under partial information

find a maximum chain $C \subseteq P$
while $P - C \neq \emptyset$ **do**
  remove an element of $P - C$ and insert it in C with a binary search
end while
return C

[4] We remark that this form is very similar to traditional channel-coding/entropy results, relating the log volume of the "typical set" of errors to the entropy nH of the errors. Even in this poset setting, our proofs used "volume" as an intermediary – but perhaps, the similarities are only superficial.

[5] Can be done in linear time, by finding the longest path in the DAG.

6.2 Analysis

We insert $|P - C| = n - |C|$ elements into C, and each insertion requires at most $\log n$ comparison queries (for binary search). We want to relate this number of comparisons to the lower-bound $\log e(P)$. We just need the results from above, showing that if all chains are short, then the entropy $H(\overline{P})$ is high, and hence so is $e(P)$. Formally, Lemma 7 says $|C| \geq n2^{-H(\overline{P})}$. Then the number of comparisons is at most:

$$\begin{aligned} \log n\ (n - |C|) &\leq \log n(n - n2^{-nH(\overline{P})}) && \text{(Lemma 7)} \\ &\leq \log n\ (\ln(2)\ nH(\overline{P})) && \text{(plot bound)} \\ &= O(\log n \cdot \log e(P)) && \text{(Theorem 2)} \end{aligned}$$

Therefore, Algorithm 1 has runtime $O(n^2)$ and query complexity $O(\log n \cdot \log e(P))$. This is within a log factor of the optimal query-complexity $\log e(P)$.

7 Merge Sort

We now introduce a more efficient sorting algorithm whose design follows from merge sort. To fully understand this algorithm, it requires a handful of additional proofs that provide limited new intuition, and as such we will merely summarize their results. The merge sort first decomposes the given poset into several long chains. Then it merges these chains starting from the smallest to largest. We can bound the number of comparisons using arguments relating this process to Huffman trees.

This argument is interesting because it uses both kinds of entropy: graph-entropy (for arguing about chains in posets), and classical entropy (for arguing that Huffman-type merging is optimal).

7.1 Algorithm

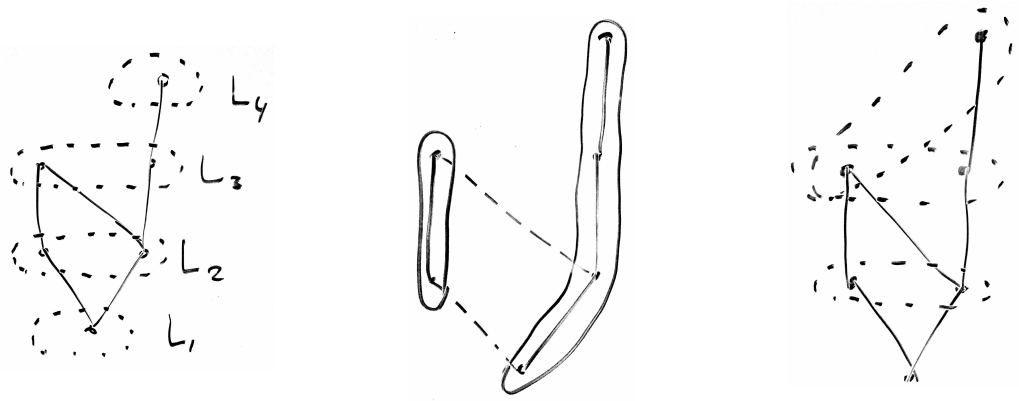
This is the natural extension of insertion sort to merge sort: Instead of finding one maximal chain and merging all other elements into it, we find many long chains, and merge them together.

Algorithm 2 (From [1]). Mergesort-like algorithm for sorting under partial information.

find a greedy chain decomposition $C_1, \ldots, C_k$ of P. (See Section 7.2).
$\mathcal{C} \leftarrow \{C_1, \ldots, C_k\}$
while $|\mathcal{C}| > 1$ **do**
  pick the two smallest chains C and C' in $\mathcal{C}$
  merge C and C' into a chain C'', in linear time
  remove C and C' from $\mathcal{C}$, and replace them by C''
end while
return the unique chain in $\mathcal{C}$

7.2 Greedy chain decomposition

The first step of the greedy algorithm is to arrange the poset into a number of levels where level L_1 represents all the elements which are not "greater" then any other. Then level L_i will represent all elements that are bigger then elements in the L_{i-1} level. An example is shown in Figure 7a. We then find maximal length chains from our level sets as seen in Figure 7b.



(a) Poset broken down into levels. (b) Selected chains from levels. (c) Select maximal antichains for our poset.

Figure 7: Poset chains construction visualized.

Using this method for greedily creating chains allows us to relate the number of chains created to size of an antichain (an independent set, or set of elements that can not be compared). This intuitively makes since as each chain will have exactly one element in a maximal antichain.

7.3 Analysis

First we will analysis the query complexity for a given $\{C_1, ..., C_k\}$ which represent ordered sets that were generated using the greedy algorithm as seen in Figure 8.

Lemma 10 *Given the ordered sets $\{C_1, ..., C_k\}$, let g be the entropy of the probability distribution $\{\frac{|C_1|}{n}, ..., \frac{|C_k|}{n}\}$. Then the query complexity of merging $\{C_1, ..., C_k\}$ will be $(1+g)n$.*

Proof. As seen in algorithm 2 we are merging the the two smallest elements during each iteration of the algorithm. This is akin to generating a Huffman tree where the probability of a given symbol is given by $\frac{|C_i|}{n}$, this means that the expected root-to-leaf distance will be $\leq g + 1$.

Now to bound the number of comparisons, we consider the merging step at each internal node. To merge the ordered sets C and C' this will require at most $|C| + |C'|$ comparisons, therefore we can "assign" one comparison to each element in our set. Each element will then have an average number of comparisons less then $1 + g$, so we can say that the total number of comparisons is $(1 + g)n$.

More formally let t_i be the length of a path from the root to a leaf, then:

$$\sum_i t_i |C_i| = n \sum_i t_i \frac{|C_i|}{n} \leq (g+1)n$$

□

We then need to show that the greedy chain decomposition $\{C_i\}$ used by the algorithm is a "good" distribution (ie, has entropy not much higher than the optimal $\log e(P)$). We can do this using Theorem 3, which relates the entropy of the greedy chain distribution to the graph-entropy $H(\overline{P})$ (which we already know is related to $\log e(P)$):

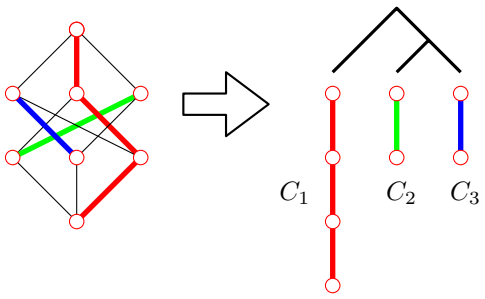


Figure 8: Greedily generating ordered subsets from a given poset. Figure from [1].

Theorem 3 *Let P be a poset on n elements, and let g be the entropy of the distribution $\{\frac{|C_i|}{n}\}$ generated by a greedy chain decomposition of P. Then:*

$$g \leq O(H(P)$$

The full proof for (a more precise version of) this theorem can be found in [10].

Now we can show that the query complexity (at most $(1+g)n$ by Lemma 10) is:

$$\begin{aligned} (1+g)n &\leq O(nH(\overline{P}) + n) && \text{(Theorem (3))} \\ &\leq O(\log e(P) + n) && \text{(Theorem (2))} \end{aligned}$$

Therefore, the query complexity of merge-sort is $O(\log e(P) + n)$. This is only an additive factor away from the optimal $\log e(P)$. [6]

References

[1] J. Cardinal, S. Fiorini, G. Joret, R. M. Jungers, and J. I. Munro, "Sorting under partial information (without the ellipsoid algorithm)," *CoRR*, vol. abs/0911.0086, 2009. [Online]. Available: http://arxiv.org/abs/0911.0086

[2] R. Stanley, "Two poset polytopes," *Discrete & Computational Geometry*, vol. 1, no. 1, pp. 9–23, 1986. [Online]. Available: http://dedekind.mit.edu/~rstan/pubs/pubfiles/66.pdf

[3] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. [Online]. Available: http://www.mast.queensu.ca/~math474/shannon1948.pdf

[4] A. Rao, "CSE533: Information theory in computer science, lecture 4," 2010. [Online]. Available: http://homes.cs.washington.edu/~anuprao/pubs/CSE533Autumn2010/lecture4.pdf

[5] M. Cheraghchi, "Information theory and applications in tcs. lecture 14: Graph entropy," 2013. [Online]. Available: http://www.cs.cmu.edu/~venkatg/teaching/ITCS-spr2013/notes/lect-mar19.pdf

[6] ——, "Information theory and applications in tcs. lecture 15: Applications of graph entropy," 2013. [Online]. Available: http://www.cs.cmu.edu/~venkatg/teaching/ITCS-spr2013/notes/lect-mar21.pdf

[6] The more careful analysis given in [1] shows the scaling is roughly $(1+\varepsilon)\log e(P) + O(\log\left(\frac{1}{\varepsilon}\right))n$.

[7] G. Simonyi, "Graph entropy: a survey," *Combinatorial Optimization*, vol. 20, pp. 399–441, 1995. [Online]. Available: http://www.renyi.hu/~simonyi/grams.pdf

[8] I. Csiszár, J. Koerner, L. Lovasz, K. Marton, and G. Simonyi, "Entropy splitting for antiblocking corners and perfect graphs," *Combinatorica*, vol. 10, no. 1, pp. 27–40, 1990. [Online]. Available: http://www.renyi.hu/~simonyi/csklms.pdf

[9] J. Kahn and J. H. Kim, "Entropy and sorting," *Journal of Computer and System Sciences*, vol. 51, no. 3, pp. 390–399, 1995. [Online]. Available: http://research.microsoft.com/en-us/um/redmond/groups/theory/jehkim/papers/entropy.pdf

[10] J. Cardinal, S. Fiorini, G. Joret, R. M. Jungers, and J. I. Munro, "An efficient algorithm for partial order production," *CoRR*, vol. abs/0811.2572, 2008. [Online]. Available: http://arxiv.org/abs/0811.2572