Reconstructing Partial Orders from Linear Extensions

Proceso L. Fernandez[†], Lenwood S. Heath[*],
Naren Ramakrishnan[*], and John Paul C. Vergara[†]
[†] Department of Information Systems and Computer Science,
Ateneo de Manila University, Quezon City 1108, Philippines
[*] Department of Computer Science, Virginia Tech, Blacksburg VA 24061, USA

ABSTRACT

Reconstructing system dynamics from sequential data traces is an important algorithmic challenge with applications in computational neuroscience, systems biology, paleontology, and physical plant engineering. Here, we formalize a key computational task in network reconstruction, namely recovering complex order-theoretic constraints among the system variables underlying a given dataset. Specifically, we focus on the problem of reconstructing partial orders (posets) from their linear extensions. We discuss the theoretical complexity of this problem, a general framework to pose and study various inference tasks, and sketch algorithmic results for mining restricted classes of posets.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining; I.2.6 [Artificial Intelligence]: Learning

General Terms: Algorithms.

Keywords: partial orders, posets, linear extensions.

1. INTRODUCTION

The problem of reconstructing system dynamics from sequential data traces is an important one, with applications in computational neuroscience [5], systems biology [1, 10], paleontology [9], and physical plant engineering [4]. In these applications, we are given time-indexed discrete symbol sequences or continuous-valued measurements, and the aim is to recover an underlying system-wide network model (reflecting connectivity, hierarchy, or strength of influences) of the observed temporal data. A key step in such network reconstruction is to elucidate order-theoretic constraints (e.g., lag, lead, or lack thereof) among the system variables underlying a given dataset.

In neuroscience, for instance, the goal is to ascertain the connectivity of the neuronal network from sequential information (time stamps, delays) about individual neuron firings. In systems biology, researchers seek to reconstruct an underlying reaction pathway by studying correlations between enzyme concentrations and protein levels. In paleontology, the approach is to infer evolutionary relationships between various taxa, in particular whether evidence supports that some species definitely originated (or not) before another species. Finally, in physical plant engineering, the data comprises symbol sequences indicative of process stages and diagnostics, and the goal is to identify causative relationships that might precede an event of interest.

Network reconstruction algorithms for unraveling system dynamics fall into two main categories. In the first category, we assume a generative model for data and seek to infer parameters of this model, conditioned on observed data. This is typically approached probabilistically , e.g., using ML or MAP estimation. In the second category, we identify constraints inferable from the given data and attempt to piece together these constraints into a system-wide model that summarizes, reconciles, or compresses the individual constraints.

Here, we formalize problems that require the inference of order constraints from sequential data to reconstruct partial order information, in particular, to infer partial orders (posets) from linear extensions. Mannila and Meek [6] studied a version of these problems; they cast it in a probabilistic setting and also presented algorithms to mine a specific category of posets. We carefully study the theoretical complexity of this problem, present a general framework to pose and study various inference tasks, and algorithms for mining restricted classes of posets.

2. PRELIMINARIES

Let V be a finite set of cardinality $n \geq 0$. A *partial order or poset* on V is a binary relation $P \subseteq V \times V$ that is reflexive, transitive, and antisymmetric. For any $u, v \in V$, we write $u \leq_P v$ (or simply $u \leq v$ when P is clear) if $(u, v) \in P$ and $u <_P v$ (or simply $u < v$ when P is clear) if $u \leq_P v$ and $u \neq v$. The *rank* $\mathrm{rank}(v; P)$ of element v in poset P is 1 plus the number of elements less than v:

$$\mathrm{rank}(v; P) \quad = \quad 1 + |\{u \in V \mid u <_P v\}|.$$

Poset P is a *total order* on V if, for all $u, v \in P$, either $u \leq v$ or $v \leq u$. A total order T on P determines a unique n-tuple $(v_1, v_2, \ldots, v_n)$ of the elements of V such that $v_1 < v_2 < \cdots < v_n$; we employ this tuple notation when we need the explicit order of elements. Note that ranks are unique in total orders; if $T = (v_1, v_2, \ldots, v_n)$, then $\mathrm{rank}(v_i; T) = i$. Let P be a poset on V. A *linear extension* of P is a total order L on V such that $P \subset L$; $\mathcal{E}(P)$ is the set of all linear extensions of P. We say that P *generates* $\mathcal{E}(P)$.

Let P be a poset on V and let $u, v \in V$. The element v

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

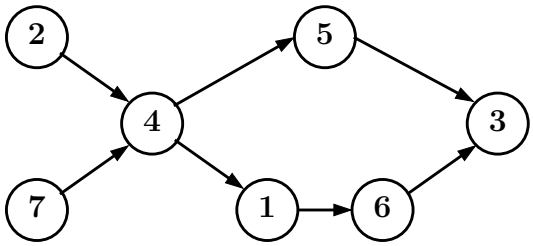

Figure 1: Hasse diagram of the example poset

covers the element u in P if $u < v$ and there is no $w \in P$ such that $u < w < v$. It is easy to prove that, if v covers u in P, then there is some linear extension of P in which u and v are immediately adjacent. The *Hasse diagram* $H(P)$ of P is the directed acyclic graph $G = (V, A)$ have vertex set V and arc set $A = \{(u, v) \mid v \text{ covers } u\}$. Alternately, the arc set of the Hasse diagram of P is the smallest binary relation on P whose transitive closure is P.

As an example, let $V = \{1, 2, 3, 4, 5, 6, 7\}$, and let

$$\begin{aligned} P \;=\; & \{(1,6),(1,3),(2,1),(2,3),(2,4),(2,5),(2,6),\\ & (4,1),(4,3),(4,5),(4,6),(5,3),(6,3),\\ & (7,1),(7,3),(7,4),(7,5),(7,6)\} \end{aligned}$$

be a binary relation on V. The reader may verify that P is a poset on V. Figure 1 illustrates a Hasse diagram of a poset and the linear extensions it generates.

The set of linear extensions of P is readily computed to be

$$\begin{aligned} \mathcal{E}(P) \;=\; & \{(2,7,4,1,5,6,3),(7,2,4,1,5,6,3),\\ & (2,7,4,1,6,5,3),(7,2,4,1,6,5,3),\\ & (2,7,4,5,1,6,3),(7,2,4,5,1,6,3)\}, \end{aligned}$$

where the six linear extensions are given in tuple notation.

Much attention has been given to the combinatorial problems of counting and generating the linear extensions of a given poset. Brightwell and Winkler [2] prove that the problem of determining the number of linear extensions of a given poset is #P-complete. Pruesse and Ruskey [8] provide an algorithm that generates all linear extensions of a given poset, which may be exponential in n in number.

3. PROBLEM DEFINITIONS

In this paper, we investigate problems whose input is a set Υ of total orders on a fixed base set V. The problem space that we have in mind results in a poset that generates (or approximately generates) Υ, in the senses we develop below. The simplest nontrivial problem from the problem space asks whether there is a single poset that generates the input set.

Generating Poset

INSTANCE: A set Υ of total orders on $V = \{1, 2, \ldots, n\}$.

QUESTION: Is there a poset P on V that generates Υ, that is, such that $\mathcal{E}(P) = \Upsilon$?

A *poset cover* for a set Υ of total orders on V is a set $\mathcal{P}$ of posets such that the union of all linear extensions of all posets in $\mathcal{P}$ is Υ, that is, such that $\Upsilon = \bigcup_{P \in \mathcal{P}} \mathcal{E}(P)$. There is always at least one poset cover of Υ, since Υ is a poset cover of itself. The computationally interesting problem is to minimize the number of posets in a poset cover.

Poset Cover

INSTANCE: A nonempty set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of total orders on $V = \{1, 2, \ldots, n\}$.

SOLUTION: A poset cover $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ of Υ such that k is minimum.

Heath and Nema [3] have recently proved that Poset Cover is NP-complete. Hence, to investigate polynomial-time solvable variants of Poset Cover, we restrict our attention to particular classes of posets and poset covers whose elements come from a particular class. Let C be a predicate applicable to posets (perhaps C characterizes the Hasse diagram of a poset). A poset on V that satisfies C is called a C-poset. Each such predicate defines this class of posets: $\{P \mid P \text{ is a } C\text{-poset}\}$.

We define two problems for each such predicate C.

Generating C-Poset (GENPOSET_C)

INSTANCE: A nonempty set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of total orders on $V = \{1, 2, \ldots, n\}$.

QUESTION: Is there a C-poset P on V that generates Υ, that is, such that $C(P)$ is true and $\mathcal{E}(P) = \Upsilon$?

C-Poset Cover (COVER_C)

INSTANCE: A nonempty set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of total orders on $V = \{1, 2, \ldots, n\}$.

SOLUTION: A poset cover $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ of Υ such that k is minimum and $C(P_i)$ is true for every $P_i \in \mathcal{P}$.

As a simple example, let PATH be the predicate that describes a poset whose Hasse diagram is a simple path. There is exactly one linear extension associated with a PATH-poset, namely, itself. Hence, both $\text{GENPOSET}_\text{PATH}$ and COVER_PATH are easily solved in polynomial time.

4. ALGORITHMS FOR MINING PARTIAL ORDERS

Here, we sketch some algorithmic results for particular problems in our problem space.

First, the Generating Poset problem can be solve in time polynomial in n, the cardinality of V, and m, the cardinality of Υ, as we show in the following theorem.

Theorem 1. *The decision problem* Generating Poset *can be solved with an $O(mn^2)$-time algorithm that takes a set of total orders Υ as input and finds the poset P that generates Υ, if it exists.*

Proof. The correctness and time-complexity of the Generating Poset algorithm in Figure 2 proves the theorem. The time complexity of Generating Poset is clear from the comments in the pseudocode. It remains to show that the result returned by Generating Poset is the correct one.

If there is no poset that generates Υ, then Generating Poset will return failure, which is correct. Otherwise, let ρ be any poset cover of Υ. Since $\rho \subset L_i$, for each i, we must have $\rho \subset \bigcap_{i=1}^{m} L_i = P$.

To obtain a contradiction, assume that there are $x, y \in V$ such that $x \not<_\rho y$ while y covers x in P. Let $L_i \in \Upsilon$ be such that x and y are adjacent in L_i, that is, in tuple notation, we may write $L_i = (v_1, v_2, \ldots, v_s, x, y, v_{s+3}, \ldots, v_n)$. Let $L_i' = (v_1, v_2, \ldots, v_s, y, x, v_{s+3}, \ldots, v_n)$. Then, $L_i' \notin \Upsilon$, since every total order in Υ has $x < y$. But, L_i' is a linear extension of ρ. Since ρ is a poset cover of Υ, we have

ALGORITHM: Generating Poset

INPUT: A set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of total orders on $V = \{1, 2, \ldots, n\}$.

OUTPUT: A poset P on V such that $\mathcal{E}(V) = \Upsilon$, if one exists.

First, compute the intersection of the total orders. Let

$$P = \bigcap_{i=1}^{m} L_i,$$

the largest poset on V that has every L_i as a linear extension. P can be constructed in $O(mn^2)$, since there are $O(n^2)$ elements in each L_i.

Now, check the cardinality of $\mathcal{E}(P)$. Use the algorithm of Pruesse and Ruskey [8] to generate the linear extensions of P in $O(n)$ amortized time per total order generated. If that algorithm ever generates an $m+1^{\text{st}}$ total order, then there is a total order in $\mathcal{E}(P) - \Upsilon$; in that case, abort the generation of linear extensions and return failure. Otherwise, the algorithm will terminate after generating precisely the m total orders in Υ. In that case, return P. In either case, verifying whether $\mathcal{E}(P) = \Upsilon$ can be done in $O(mn)$ time.

Figure 2: Polynomial-time algorithm to solve Generating Poset

$L'_i \in \Upsilon$, a contradiction. We conclude that $\rho = P$. The theorem follows. □

We now restrict our attention to a particular class of posets. In a poset, a *hammock* is a triple (u, S, v) such that $u, v \in V$, $S \subset V$, $|S| > 1$, and

$$\begin{aligned} S &= \{w \in V \mid w \text{ covers } u\} \\ &= \{w \in V \mid v \text{ covers } w\}; \end{aligned}$$

in this case, we say that the hammock (u, S, v) is *determined* by u and v. We also say that the subgraph of the Hasse diagram of P generated by $\{u, v\} \cup S$ is a *hammock*. Consider the Hasse diagram in Figure 4. Vertices 3 and 7 determine the hammock $(3, \{4, 14\}, 7)$. The figure, in fact, contains three hammocks, namely, $(3, \{4, 14\}, 7)$, $(6, \{9, 12\}, 11)$, and $(11, \{2, 8, 13\}, 10)$.

Poset P is a *hammock-poset* if there exists a partition

$$V_1, V_2, \ldots, V_s$$

of V that satisfies the following properties:

1. $|V_1| = |V_s| = 1$;
2. For every i satisfying $1 \leq i < s$, either $|V_i| = 1$ or $|V_{i+1}| = 1$;

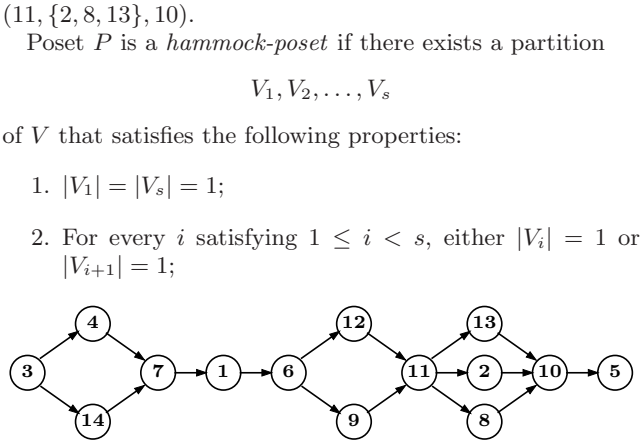

Figure 3: Hasse diagram of a hammock-poset

3. If $|V_j| > 1$, $V_{j-1} = \{u\}$, and $V_{j+1} = \{v\}$, then (u, V_j, v) is a hammock in P;
4. If $|V_j| = |V_{j+1}| = 1$, $V_j = \{u\}$, and $V_{j+1} = \{v\}$, then $V_j = \{w \in V \mid v \text{ covers } w\}$ and $V_{j+1} = \{w \in V \mid w \text{ covers } u\}$.

The Hasse diagram of a hammock-poset is a *hammock chain*. The Hasse diagram in Figure 4 is a hammock chain, with partition

$$\{3\}, \{4, 14\}, \{7\}, \{1\}, \{6\}, \{12, 9\}, \{11\}, \{2, 8, 13\}, \{10\}, \{5\}.$$

In a hammock chain, a *link vertex* is a vertex in a singleton V_i, while a *hammock vertex* is a vertex in a partition block V_i containing more than one element. In Figure 4, the link vertices are 3, 7, 1, 6, 11, 10, and 5, while the hammock vertices are 4, 14, 12, 9, 13, 2, and 8. Note that every cutpoint in the Hasse diagram is a link vertex.

Let HAMMOCK be the predicate that is true only for hammock-posets. We find that the Generating Poset problem can be solved more efficiently for hammock-posets than in the general case of Theorem 1.

Theorem 2. *There is an $O(nm + n^2)$-time algorithm to solve* GENPOSET$_{\text{HAMMOCK}}$.

Proof. The correctness and time-complexity of the Generating Hammock-Poset algorithm in Figure 4 proves the theorem. The time complexity is clear from the comments in the pseudocode. The proof of the correctness of the algorithm will be presented in a later publication. □

Most sets of total orders do not have a corresponding generating poset. Hence, the Poset Cover problem is usually the one that must be addressed. Let Υ be a set of total orders on V. A poset P on V is *feasible* if $\mathcal{E}(P) \subseteq \Upsilon$, that is, if every linear extension of P is one of the total orders. Any solution to the Poset Cover problem must consist of one or more feasible posets for the input set Υ. The number of feasible posets for Υ may be exponential in m. However, if we restrict our attention to some particular classes of posets, it may be possible to show that the number of feasible posets in that class is polynomial in m and indeed can be generated in polynomial time.

Let KITE be the predicate that is true for a hammock-poset that contains exactly one hammock; call such a poset a kite-poset. For $k > 1$, let KITE(k) be the predicate that is true for a kite-poset whose single hammock has exactly k hammock vertices. The set of feasible kite-posets can be generated in polynomial time.

Theorem 3. *Let $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ be a nonempty set of total orders on $V = \{1, 2, \ldots, n\}$. The set of all feasible kite-posets for Υ can be generated in $O(n^2(nm \log m))$ time.*

Proof. Suppose that P is a feasible kite-poset of Υ. Then, there exist two integers i and j such that $1 \leq i < j \leq n$, $j - i \geq 3$, there is a unique element $u \in V$ of rank i, there is a unique element $v \in V$ of rank j, and (u, S, j) is the unique hammock of P with $S \subset V$ having cardinality $j - i - 1$. There are only $O(n^2)$ possible i, j pairs to consider.

Fix one i, j pair satisfying $1 \leq i < j \leq n$ and $j - i \geq 3$. If $L = (v_1, v_2, \ldots, v_n)$ is a total order on V, then define its i, j-restriction to be

$$L(i, j) = (v_1, v_2, \ldots, v_{i-1}, v_i, v_j, v_{j+1}, \ldots, v_n).$$

ALGORITHM: Generating Hammock-Poset

INPUT: A set $\Upsilon = \{L_1, L_2, \ldots, L_m\}$ of total orders on $V = \{1, 2, \ldots, n\}$.

OUTPUT: A hammock-poset P on V such that $\mathcal{E}(V) = \Upsilon$, if one exists.

First, determine the rank frequency of Υ. Let $F = (f_{ij})$ be an $n \times n$ frequency matrix, where f_{ij} is the count of the number of total orders in Υ for which the rank of $i \in V$ is j. Note that every row of F sums to m. Frequency matrix F can be constructed in $O(mn)$ time by iterating through each of the n elements of the m total orders and incrementing the appropriate count.

Second, classify every vertex as a link or hammock vertex. Consider a single vertex $i \in V$. Let $j_1, j_2, \ldots, j_k$ be the ranks for which $f_{i,j_r} > 0$. Without loss of generality, assume that $j_1 < j_2 < \cdots < j_k$. If $k = 1$, then $f_{i,j_1} = m$, and we classify vertex i as a link vertex of rank j. Otherwise, every $f_{i,j_r} < m$. If k does not divide m, then return failure. If it is not true that every $f_{i,j_r} = m/k$, then return failure. If there exists r, with $1 \leq r < k$, such that $j_{r+1} - j_r \neq 1$, then return failure. Otherwise, classify vertex i as a hammock vertex with rank j_1. Classification can be done in $O(n^2)$ time.

Third, construct a candidate poset P. For $i, j \in V$ with $i \neq j$, let ρ_i and ρ_j be the ranks assigned above. Then, $i <_p j$ if $\rho_i < \rho_j$ as integers. Clearly, P can be constructed in $O(n^2)$ time.

Finally, check that P is a hammock-poset and generates Υ. If so, return P. If not, return failure. This step can be accomplished in $O(n^2 + mn)$ time.

Figure 4: A polynomial-time algorithm to solve $\text{COVER}_{\text{HAMMOCK}}$

Sort the elements of Υ by their i, j restrictions. This can be done in $O(mn \log m)$ time using merge sort. Let $L_r \in \Upsilon$. There is a feasible kite-poset that has linear extension L_r if and only if there are $(j - i - 1)!$ elements of Υ having the same i, j restriction as L_r. This is easily determined by scanning the sorted total orders in $O(mn)$ time.

The computation requires $O(n^2(nm \log m))$ time. The theorem follows. □

In fact, if we restrict our attention to KITE(2)-posets, the Poset Cover problem is solvable in polynomial time.

Theorem 4. *There is an $O(m^{1.5}n + n(nm \log m + mn))$-time algorithm to solve* $\text{COVER}_{\text{KITE(2)}}$.

Proof. For a set Υ of total orders on V, the set of all feasible posets that satisfy the predicate KITE(2) can be generated in $O(n(nm \log m))$ time by restricting the algorithm of Theorem 3 to i, j pairs such that $j - i = 3$. Let p be the number of feasible posets returned; clearly, $p = O(mn)$, since every total order is associated with $n - 1$ kite-posets satisfying KITE(2). Construct an undirected graph with vertex set Υ and an edge between L_r and L_s if one of the generated posets has both L_r and L_s as linear extensions. This graph has m vertices and p edges. We can find a maximum matching in the graph using the algorithm of Micali and Vazirani [7], which runs in $O(m^{1/2}p) = O(m^{1.5}n)$ time. Choosing the kite-poset for each of the edges in a maximum matching plus one edge for every unmatched vertex yields an optimal solution to $\text{COVER}_{\text{KITE(2)}}$. □

5. CONCLUSIONS

This short paper has briefly formalized problems related to identifying sets of posets that summarize or compress order-theoretic data sets. Through formalization, we hope to open the door for greater research into these problems. While the problems bear much resemblance to classical set cover problems, they also have striking differences, as the objects to be used in a solution are only available implicitly, rather than explicitly given as in set cover problems. There are also variations of Poset Cover that ask for approximate solutions. For example, one might allow a solution that is a set of posets that has linear extensions outside of the input set of total orders; in this case, one must decide what it means to have a good approximation.

6. REFERENCES

[1] A. Arkin, P. Shen, and J. Ross. A Test Case of Correlation Metric Construction of a Reaction Pathway from Measurements. *Science*, Vol. 277(5330):pages 1275–1279, Aug 1997.

[2] G. Brightwell and P. Winkler. Counting Linear Extensions. *Order*, Vol. 8(3):pages 225–242, 1991.

[3] L.S. Heath and A.K. Nema. Poset Cover is NP-Complete. In preparation for submission, 2006.

[4] S. Laxman, P.S. Sastry, and K.P. Unnikrishnan. Discovering Frequent Episodes and Learning Hidden Markov Models: A Formal Connection. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17(11):pages 1505–1517, 2005.

[5] A. K. Lee and M. A. Wilson. A Combinatorial Method for Analyzing Sequential Firing Patterns Involving an Arbitrary Number of Neurons Based on Relative Time Order. *Journal of Neurophysiology*, Vol. 92(4):pages 2555–2573, 2004.

[6] H. Mannila and C. Meek. Global Partial Orders from Sequential Data. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 161–168, 2000.

[7] P.A. Peterson and M.C. Loui. The General Maximum Matching Algorithm of Micali and Vazirani. *Algorithmica*, Vol. 3:pages 511–533, 1988.

[8] G. Pruesse and F. Ruskey. Generating Linear Extensions Fast. *SIAM Journal on Computing*, Vol. 23(2):pages 373–386, 1994.

[9] K. Puolamaki, M. Fortelius, and H. Mannila. Seriation in Paleontological Data: Using Markov Chain Monte Carlo Methods. *PLoS Computational Biology*, Vol. 2(2), Feb 2006.

[10] C.H. Wiggins and I. Nemenman. Process Pathway Inference via Time Series Analysis. *Experimental Mechanics*, Vol. 43(3):pages 361–370, Sep 2003.