Graph labeling schemes

Antoine Amarilli

Abstract

We present the problem of finding efficient labeling schemes to encode the reachability relation represented by a DAG. We focus on the well-studied cases of interval labeling and interval-containment labeling, and spell out the details of an interval-containment labeling algorithm. We then look at existing relaxations of these schemes with non-constant label sizes, and present ideas for a relaxation scheme on interval-containment orders.

1 Introduction

General problem. We are given a DAG G on which we will have to answer reachability queries: can we reach node a from node b in G? Which preprocessing can we do on this DAG to answer those queries efficiently?

Without preprocessing, the reachability can be tested by a simple graph exploration which is $O(E)$ in the worst case. If we precompute the answer to each reachability query (which is akin to computing the transitive closure), we can answer all queries in $O(1)$, but need $O(V^2)$ memory. Can we do better? The well-known scheme of [ABJ89] is efficient in practice but still requires $O(V^2)$ memory in the worst case. In particular, can we devise a scheme which requires memory $O(V)$ while allowing us to answer reachability queries in $O(1)$?

An example. Consider the graph G in figure 1. We will encode the transitive closure of G by representing each node as an interval, so a total representation size of $O(1)$), and ensuring that for any two nodes x and y, y is reachable from x if and only if the interval representing y is included in the interval representing x.

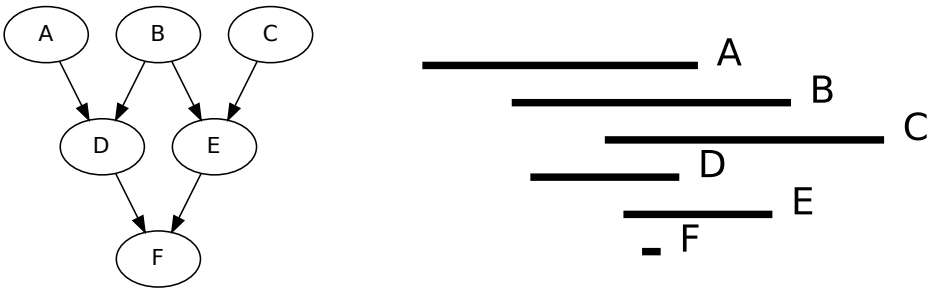

Figure 1: An example graph G and intervals.

Each interval is represented as two values, so the storage cost per node is $O(1)$ (we consider that the values used fit in a machine word, which is true for graphs of reasonable size). Hence, the total storage cost is $O(V)$, and testing if one interval is included in the other costs $O(1)$. We will present this scheme (the *interval-containment order*) in the rest of this document. Sadly, not all graphs can be represented by intervals in this way.

Labeling schemes. The scheme presented above is an example of a *labeling scheme*: attach an constant-sized label $i(x)$ to each node x, define a constant-time boolean function f on couple of labels, and require that $f(i(x), i(y))$ iff y is reachable from x in G. More specifically, let us focus on the case where we set a given constant k, require the labels are k-uples of integer values, and require f to be a boolean combination of the predicates $i(x)_i < i(y)_j$ and $i(y)_i < i(x)_j$ for $(i, j) \in \{1, ..., k\}^2$. In other words, the partial order on vertices defined by the labels is some combination of the total orders on the possible couple of label values, and we require this partial order to match the order encoded by the reachability relation in our DAG. The example presented above is a labeling scheme in this sense, for $k = 2$.

How expressive are those labeling schemes, depending on k? If we take $k = 1$, it is clear that the only orders that can be represented by such a scheme are linear orders (matching the linear order on labels) over groups of pairwise incomparable nodes (which carry the same label). If we take $k = 2$, we can see the values as the left and right endpoints of intervals: if we set f to be the interval precedence order or the interval containment order, the class of orders that can be represented are the so-called interval orders and interval-containment orders that we will describe later. Interestingly, these orders have different expressivity and different recognition and labeling algorithms. I do not know if different choices of f for $k = 2$ would give rise to different classes.

For arbitrary values of k, the behaviour of the product order (boolean "and"), corresponding to $f(i(x), i(y)) = \bigwedge_{j=0}^{k} i(x)_j < i(y)_j$, is well-known: the smallest k for which a given partial order admits such a representation is its *dimension*, and the $(i(x)_j)_j$ are linear extensions of the order. The interval-containment order is actually the product order for $k = 2$. Other well-studied classes of orders for arbitrary values of k are containment relations of various kinds of geometrical structures [FT99]. A natural question is to know whether the expressivity increases indefinitely with k (ie. for every k one can devise an order which can not be represented using k values), or if some value k_0 is sufficient to represent all possible orders (ie. the hierarchy collapses at k_0). It is known that this hierarchy does not collaspe if we restrict f to be the product order, but I do not know what happens for arbitrary values of k: it seems very likely that it does not collapse but I'm not sure of how to prove it.

A related notion is defined in the conclusion of [GNT90], under the name *local presentation*, with the minor difference that the boolean function can also test equality in addition to inequality. It does not seem to have been studied since.

Relaxations. From the comparison functions which can work on a certain class of graphs, a natural relaxation is to extend the comparison to a sequence of labels, and to devise a labeling scheme which is correct and which minimizes

the total number of labels used. This approach is used by [ABJ89] with an quasi-optimality proof for their labeling scheme, and a greedy non-optimal approach is used by [Cap94]. Of course, such approaches are less expressive than using larger labels (because their comparison functions are a simple expression of comparison functions on smaller labels), but they are more fine-grained because they make it possible to use a different number of values on different nodes.

Outline. In the rest of this text, we present the vocabulary of graphs and transitive orders in section 2. We present interval orders and interval-containment orders in section 3. We conclude by studying relaxations of interval orders and interval-containment orders in section 4.

2 Vocabulary

2.1 Graphs

Consider a DAG $G = (V, E)$. The *transitive closure* of G is $G^* = (V, E^*)$ where E^* is the transitive closure of E seen as a binary relation. The *transitive reduction* of G is the smallest DAG G' such that $G'^* = G'$; it is unique.

We say that a DAG $G = (V, E)$ is *transitive* if for all $x, y, z \in V$, $(x, y) \in E (y, z) \in E \Rightarrow (x, z) \in E$. Equivalently, G is transitive iff $G = G^*$. We identify transitive DAGs with the strict partial order that they define on V: $x < y$ iff $(x, y) \in E$. Hence, we can say that any DAG G defines a strict partial order on V, which is the one represented by its transitive closure G^*.

Given a DAG $G = (V, E)$, we write $E^- = \{(y, x) \in V^2 | (x, y) \in E\}$ and $G^- = (V, E^-)$ the *reverse* of G. We write $G^\sim = (V, \{\{x, y\} | (x, y) \in E\})$ the undirected graph obtained by forgetting the orientation of the edges of G. We define the *complement* of an undirected graph G as $\overline{G} = (V, \{\{x, y\} \in V^2 | \{x, y\} \notin E\})$ and the complement of a DAG G to be the complement of $G^\sim$.

2.2 Orders

A *strict partial order* $<$ on a set X is an antisymmetric transitive relation on X. A *total order* is a strict partial order $(X, <)$ such that for every $x, y \in X$, either $x = y$ or $x < y$ or $y < x$. Hence, the usual order on integers is a total order, however the order induced by the proper divisibility on positive integers ($x < y$ iff x is a proper divisor of y) is not a total order, and the orders on intervals presented in the next section are not total orders either.

We say that an order $(X_1, <_1)$ can be embedded in some other order $(X_2, <_2)$ when there is a way to encode the first order in the second, ie. a function from X_1 to X_2 which preserves order. Formally, we define an *order embedding* between two orders $(X_1, <_1)$ and $(X_2, <_2)$ as a function $f : X_1 \Rightarrow X_2$ such that for any $x, y \in X_1$, $x <_1 y$ iff $f(x) <_2 f(y)$

As explained above, a strict partial order corresponds exactly to a transitive DAG with X as vertices and $(x, y) \in E$ whenever $x < y$. The *incomparability graph* of a partial order $(X, <)$ represented by a transitive graph G is defined as $\overline{G}$, the undirected graph of incomparable pairs of $<$.

3 Interval labeling schemes

We present here the two well-known classes of interval labeling schemes: interval-containment orders and interval orders.

3.1 Interval-containment orders

The *strict containment relation* is a partial order on intervals of the real line ($[a, b] \subsetneq [c, d]$ iff $c < a$ and $b < d$). We say that an order is an *interval-containment order* if it can be embedded in the strict containment relation over intervals of the real line. We say that a graph is an interval-containment graph if it represents such an order. In other words, this means that given an interval-containment graph, there exists a labeling i from vertices to intervals of the real line which encodes the strict reachability relation (v is reachable from u iff. $i(v) \subsetneq i(u)$).

Given a DAG G, how can we determine if it is an interval-containment graph, and how can we determine a labeling? To do so, we must introduce more concepts from graph theory and partial order theory.

3.1.1 Transitively orientable graphs

We say that an undirected graph G is *transitively orientable* (or that it is a *comparability graph*) if there exists a DAG G' such that $G'^{\sim} = G$. A necessary and sufficient condition for G to be transitively orientable is that G does not contain an odd-length cycle with no triangular chords ([GH64], th. 1). Another necessary and sufficient condition is that G does not contain any induced subgraph from a complicated family of forbidden induced subgraphs identified by [Gal67]. Note that if G' is a transitive orientation of G, then so is G'^{-}, so, because of this symmetry, a transitively orientable graphs always admits at least two transitive orientations (and possibly a larger even number of them).

Given an undirected graph G, there is a polynomial algorithm to test if it is transitively orientable, and determine a transitive ordering [PLE71]. It proceeds as follows. We start by orienting one edge arbitrarily, which is motivated by the symmetry that we mentioned above. We then propagate the consequences of this choice by orienting all neighboring edges in the opposite direction if ordering them in the same direction would lead to a transitivity violation: we also propagate the consequences of these orientations, etc. Once all consequences have been propagated, we orient another edge arbitrarily, and we propagate. We continue to do this until all edges have been oriented, and check if the graph is transitive. This algorithm is $O(E^2)$ for the checking part, and same for the orientation part (since for each edge we potentially look at all other edges of the graph to see if we have to orient them).

Given a graph G which is not transitively orientable, it is NP-hard to determine the minimum completion of G (ie. the minimum number of edges to add to make it transitively orientable) [HMP08]. It seems that finding the minimum number of edges to remove is also NP-hard ([HSY02] allude to it and cite a book I don't have access to).

3.1.2 Order dimension

Consider $(X, <)$ a partial order. A *linear extension* of $(X, <)$ is a total order $<'$ on X such that $\forall x, y \in X, x < y \Rightarrow x <' y$. Computing a linear extension of a partial order is analogous to performing a topological sort on the transitive DAG which represents the order: you assign a distinct time $t(x)$ to every element $x \in X$ such that $x < y$ implies $t(x) < t(y)$. This uses the fact that the DAG is acyclic (ie. there are no circular dependencies). Note that we don't have $t(x) < t(y)$ implies $x < y$ unless $<$ is a total order: in other words, this is not an order embedding.

We say that a partial order $(X, <)$ is *realized* by a collection of linear orders $(<_i)_i$ if $\forall x, y \in X, x < y \Leftrightarrow \forall i, x <_i y$. In other words, $<$ is the intersection of the pairs of the $<_i$. Intuitively, total orders are represented by just one linear extension (themselves), and non-total orders must intersect disagreeing linear extensions to represent their incomparable pairs (but these linear extensions all agree on the comparable pairs, since they are linear extensions). Hence, non-total orders require at least two linear extensions to be represented, and actually we will see that two is not always enough.

The *dimension* of a partial order is the minimal size of a collection of linear orders which realizes it. Hence, a linear order has dimension 1. It is NP-complete to check if the dimension of a partial order is at most 3 [Yan82]; however, we can check in polynomial time if the dimension of a partial order is at most 2. Orders of arbitrarily high dimension can be constructed.

3.1.3 Characterization

It can be proved ([DM41], th. 3.61) that a partial order $(X, <)$ is an interval-containment order iff it is of dimension ≤ 2, that it, either it is of dimension 1 and it is a linear order, or it is of dimension 2 and it can be realized by two linear extensions P and Q. Further, we can build an interval-containment labeling of the order from these two linear extensions, by ordering the left interval endpoints according to the first linear extension and the right interval endpoints according to the second linear extension (and putting all left endpoints to the left of all right endpoints). Hence, finding an interval-containment labeling (if one exists) is equivalent to finding two linear extensions realizing the order represented by a graph.

Further, the same theorem tells us that such linear extensions exist iff the incomparability graph G of $<$ is transitively orientable. With their terminology: $<$ has a *conjugate order* $<'$ which orders the elements of the incomparability graph of $<$. Further, if we have transitive ordering $<'$ of the incomparability graph, we can build the linear extensions as follows: the first extension P is the original order $<$ extended to incomparable elements by the $<'$, and the second extension Q is $<$ extended by $<'^{-}$ (the reverse of $<'$, ie. the order relation represented by the reverse transitive ordering of the incomparability graph).

Interestingly, this means that to know if G has an interval-containment representation, one only needs to look at its comparability graph $G^{*\sim}$. In particular, this implies that G has such a representation iff G^{-} has.

A generalization of the above characterization of interval-containment orders is mentioned in [FT99] (th. 2): a partial order is a box-containment order over boxes of $\mathbb{R}^k$ (ie. Cartesian products of k intervals) iff its dimension is $\leq 2k$.

3.1.4 Simple algorithm

Using the characterization, we can build an interval-containment labeling of a graph G (or determine that none exists) in the following fashion:

1. Compute $\overline{G^*}$, the incomparability graph of the transitive closure of G.
2. Determine a transitive orientation $P = (V, E')$ (represented as a DAG) of $\overline{G^*}$ if one exists, and fail if none does.
3. Construct $G_1 = (V, E \cup E')$, and $G_2 = (V, E \cup E'^{-})$. Lemma 3.51 of [DM41] guarantees that G_1 and G_2 represent total orders (and are therefore acyclic).
4. Perform a topological sort of G_1, yielding times T_1 such that for all $x, y \in V$, if y is reachable from x in G_1 then $T_1[x] < T_1[y]$.
5. Likewise, perform a topological sort of G_2^-, yielding times T_2.
6. Assign intervals as follow: to a vertex x, assign $[T_1[x], |V| + T_2[x]]$.

The most expensive operation is the transitive orientation of the incomparability graph. The number of edges of the incomparability graph is $O(|V|^2)$, and determining a transitive orientation of it is using the algorithm that we described is thus $O(|V|^4)$, where V is the vertex set of the original graph.

3.1.5 Modular decomposition algorithms

The simple algorithm that we describe is polynomial but costly. However, it turns out that are linear time recognition algorithms for two-dimensional partial orders (apparently [MS94] and its follow-up [MS99], and [CH94]). These algorithms work by looking at the modular decomposition of the graph [Wik12a], which is a representation of the graph which generalizes somehow the idea of looking at connected components of G independently: either we split on the connected components of G, or on the connected components of G^-, or if both are connected we can partition the vertices into disjoint minimal sets ("modules") of nodes with the same neighbors and treat independently the graph up to this partition (which is a *prime graph*) and the partitions themselves. Apparently the problem of transitive orientation on a prime graph is easier. It looks like these algorithms make it possible to determine an interval labeling in quasi-linear time. I'm not sure of whether it is reasonable to implement these algorithms, and if they can really determine an interval labeling; I'm still trying to understand them.

3.1.6 Permutation graphs

We say that an undirected graph G is a *permutation graph* if G and $\overline{G}$ are both transitively orientable. Hence, if G has dimension 2 then $G^\sim$ is a permutation graph. Conversely, if $G^\sim$ is a permutation graph, then G's incomparability graph is transitively orientable and so is $G^\sim$, but since G is a DAG then we can trivially choose it as a way to order $G^\sim$. So, permutations graphs are exactly the comparability graphs of interval-containment orders. This is surprising because permutation graphs have an entirely different (but equivalent) definition as the graph of intersection of segments between points on parallel lines [Wik12b].

3.2 Interval orders

The *precedence relation* is a partial order on intervals of the real line ($[a, b] < [c, d]$ iff. $b < c$). We say that an order is an *interval order* if it can be embedded in the precedence relation over intervals of the real line.

3.2.1 Interval dimension

The interval dimension of an order is defined analogously to the order dimension but with interval order extensions instead of linear extensions. More precisely, an *interval order extension* $<'$ of a given order $<$ is an interval order such that $\forall x, y \in V, x < y \Rightarrow x <' y$. Analogously to the case of linear extensions, a partial order $(X, <)$ is *realized* by a collection of interval orders $(<_i)_i$ if $\forall x, y \in X, x < y \Leftrightarrow \forall i, x <_i y$ (ie. $<$ is the intersection of the pairs of the $<_i$). The *interval dimension* of a partial order is the minimal size of a collection of interval orders which realizes it. Hence, interval orders are the orders of interval dimension 1 (they are realized by one interval order, namely themselves).

It is known that the interval order dimension of an order is less than its order dimension ([FHM94], (1)) but that there exist orders of interval dimension 1 (aka. interval orders) of arbitrarily high dimension. In particular, there exist interval orders of dimension < 2, so some interval orders are not interval-containment orders.

Conversely, interval orders can be characterized by a forbidden pattern ([BM93], sec. 1) and this forbidden pattern itself is an interval-containment order. Therefore, some interval-containment orders are not interval orders.

This means that the classes of interval orders and interval-containment orders are different, and that neither is included in the other. Of course, some orders are in neither of those classes: for instance, take a graph with two connected components, one which is of interval dimension 2, and one which is of order dimension 3.

3.2.2 Algorithm

There is a linear time and space algorithm to recognize interval orders and obtain an interval labeling [BM93].

3.2.3 Interval graphs

An undirected graph G is an *interval graph* iff it is the graph of intersections of a family of segments of the real line. A DAG G represents an interval order iff its incomparability graph $\overline{G}$ is an interval graph. This is by definition: in an interval order, two nodes are not comparable iff their intervals intersect.

4 Relaxations

4.1 Spanning tree relaxation

In a way, [ABJ89] is a relaxation of a simple containment labeling scheme for trees. They prove that their relaxation is optimal for their scheme except for adjacent interval merging.

4.2 Interval order relaxation

The idea of extending interval orders to arbitrary DAGs by seeing them as the union of interval orders has been explored by [Cap94]. Using the algorithm of [BM93], they compute a maximal interval restriction of a DAG, which is a maximal subgraph of G which is an interval order, and greedily represent G as a union of a sequence of maximal interval restrictions. The scheme is not optimal (be it only because the maximal subgraphs are not maximum subgraphs, and because the greedy representation might not be the best one). They provide an experimental comparison of their approach to [ABJ89].

4.3 Interval-containment order relaxation

To my knowledge, there is no analogous relaxation scheme for interval-containment orders. Here, I sketch possible ideas to develop one.

Given a graph G, we can try to find a maximal subset of vertices for which we can have an interval-containment order. More precisely, we look at G^* and want to find a minimal subset of vertices V' such that we can create a graph $(G^*)'$ (not necessarily transitive or even acyclic anymore) which agrees with G^* except for couples of two vertices in V'. In the transitively orientable setting, this means that I can add or remove vertices between nodes of V' to make the resulting graph transitively orientable. Note that we do *not* propagate the results of these changes by transitivity: we do not care if the result is not transitive (or even cyclic) when elements of V' are involved, we just care about the fact that it will be correct between elements not both in V'.

We can then label with intervals which will be correct except in V'. Now, we can apply the same process to the restriction of G^* to V' and label elements of V' with a second interval only valid within V' (the first interval serves to get the correct order with vertices in $V \backslash V'$, the second to get the correct order within V'). Of course, we may need to get more than two intervals levels; when looking at two vertices labeled by two sequences of intervals, we should test the containment of the last interval of the shortest sequence with the interval at the same position in the other sequence.

A naive implementation of this idea could give something similar to that of [Cap94]: we take as many vertices as we can until we get a non-interval-containment graph, and we try again on the remaining vertices, etc. We could hope that this would experimentally yield shorter labels than [ABJ89] or [Cap94]. To implement this in practice, we would probably need a quasi-linear algorithm as the subroutine: whereas [Cap94] uses [BM93], we would probably need to use one of the complicated quasi-linear algorithms in section 3.1.5.

A harder question is: can we determine a maximum subset of vertices instead of a maximal subset of vertices? (Is this NP-complete like the other subgraph and supergraph problems on transitively orientable graphs? Is it reasonable in practice on the graphs that we are interested in?). Also, is the greedy scheme of picking the maximum subsets at each step optimal with regards to the total number of intervals or not? Have those maximal subsets any link with the modular decomposition?

Other ideas: extend [ABJ89] using an (almost) transitive orientation of the incomparability graph to guide the order in which we browse the tree. Determine

if [ABJ89]'s non-optimality is (a.) just because of the ordering or (b.) because they don't use the right spanning tree or (c.) because they use a spanning tree.

References

[ABJ89] R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. In *Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, SIGMOD '89, pages 253–262, New York, NY, USA, 1989. ACM.

[BM93] Philippe Baldy and Michel Morvan. A linear time and space algorithm to recognize interval orders. *Discrete Appl. Math.*, 46(2):173–178, October 1993.

[Cap94] Christian Capelle. Representation of an order as union of interval orders. In *ORDAL'94 (Orders, Algorithms and Applications), LNCS 831*, pages 143–162. Springer, 1994.

[CH94] Alain Cournier and Michel Habib. A new linear algorithm for modular decomposition. In *Proceedings of the 19th International Colloquium on Trees in Algebra and Programming*, CAAP '94, pages 68–84, London, UK, UK, 1994. Springer-Verlag.

[DM41] Ben Dushnik and E. W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):pp. 600–610, 1941.

[FHM94] S. Felsner, M. Habib, and R. H. Möhring. On the interplay between interval dimension and dimension. *SIAM J. Discret. Math.*, 7(1):32–40, February 1994.

[FT99] P. C. Fishburn and W. T. Trotter. Geometric containment orders: a survey. *Order*, 15(2):167–182, 1998/99.

[Gal67] T. Gallai. Transitiv orientierbare Graphen. *Acta Math. Acad. Sci. Hungar*, 18:25–66, 1967.

[GH64] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canad. J. Math.*, 16:539–548, 1964.

[GNT90] Giorgio Gambosi, Jaroslav Nešetřil, and Maurizio Talamo. On locally presented posets. *Theoretical computer science*, 70(2):251–260, 1990. `https://www.sciencedirect.com/science/article/pii/0304397590901252`.

[HMP08] Pinar Heggernes, Federico Mancini, and Charis Papadopoulos. Minimal comparability completions of arbitrary graphs. *Discrete Appl. Math.*, 156(5):705–718, March 2008.

[HSY02] S. Louis Hakimi, Edward F. Schmeichel, and Neal E. Young. Orienting graphs to optimize reachability. *CoRR*, cs.DS/0205042, 2002.

[MS94] Ross M. McConnell and Jeremy P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, SODA '94, pages 536–545, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[MS99] Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1–3):189 – 241, 1999.

[PLE71] A. Pnueli, A. Lempel, and S. Even. Transitive orientation of graphs and identification of permutation graphs. *Canad. J. Math.*, 23:160–175, 1971.

[Wik12a] Wikipedia. Modular decomposition — Wikipedia, the free encyclopedia, 2012. [Online; accessed 25-July-2012].

[Wik12b] Wikipedia. Permutation graph — Wikipedia, the free encyclopedia, 2012. [Online; accessed 23-July-2012].

[Yan82] Mihalis Yannakakis. The Complexity of the Partial Order Dimension Problem. *SIAM Journal on Algebraic and Discrete Methods*, 3(3):351–358, 1982.