On-line recognition of interval orders

Vincent Bouchitté, Roland Jégou, Jean-Xavier Rampon

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On–Line Recognition of Interval Orders

Vincent Bouchitté, Roland Jégou, Jean–Xavier Rampon

N° 2061

Octobre 1993

PROGRAMME 1

Architectures parallèles,
bases de données,
réseaux et systèmes distribués

ISSN 0249-6399

On–Line Recognition of Interval Orders

Vincent Bouchitté*, Roland Jégou**, Jean–Xavier Rampon***

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet Pampa

Rapport de recherche n ° 2061 — Octobre 1993 — 18 pages

Abstract: The first one is optimal in time and space and recognizes the transitive closure of an interval order under the suborder hypothesis which means that we add a new element to the transitive closure of an interval order and test if the new digraph is always the transitive closure of an interval order. The second one recognizes the transitive reduction of an interval order in linear space and almost linear time under the linear extension hypothesis which means that we add a new maximal element to the transitive reduction of an interval order.

(Résumé : tsvp)

*LIP, CNRS URA 1398, Ecole Normale Supérieure de Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France; email: vbouchit@lip.ens-lyon.fr
**Dept. d'Inf. Appliquée, E.N.S. des Mines de Saint Etienne, 158 Cours Fauriel, 42023 Saint Etienne Cedex 2, France; email: jegou@degas.emse.fr
***IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France; email: rampon@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 38 38 32

Reconnaissance "on–line" des ordres d'intervalles

Résumé : Dans ce papier nous présentons deux algorithmes "on–line" pour reconnaître un ordre d'intervalles. Le premier algorithme est optimal en temps et en espace et reconnaît la fermeture transitive d'un ordre d'intervalles sous l'hypothèse dite du sous–ordre: un nouvel élément est ajouté à la fermeture transitive d'un ordre d'intervalles et l'algorithme teste si la structure obtenue est la fermeture transitive d'un ordre d'intervalles. Le second algorithme est linéaire en espace et quasi–linéaire en temps, il reconnaît la réduction transitive d'un ordre d'intervalles sous l'hypothèse dite de l'extension linéaire. Autrement dit, étant donné la réduction transitive d'un ordre d'intervalles à laquelle un élément maximal est ajouté, le nouveau graphe est–il toujours la réduction transitive d'un ordre d'intervalles?

On–Line Recognition of Interval Orders

Contents

1 Introduction **2**

2 Recognition under Suborder Hypothesis **5**

3 Recognition under Linear Extension Hypothesis **13**

4 Conclusion **16**

1 Introduction

During the last decade ordered sets have been intensively investigated mainly with two points of view: the structural one and the algorithmic one, see Rival [17] [18] [19] [20] and Pouzet and Richard [16]. For many problems such as recognizing particular classes, like two dimensional orders, N–free orders, interval orders, ... (see for example the survey of Möhring [14]), and computing parameters, like dimension, jump number, bump number, ... (see for example the survey of Bouchitté and Habib [3]), proofs of intractability have been obtained or efficient sequential algorithms have been developed. Although theoretical objects of study, ordered sets have also appeared to modelize many problems encountered in scheduling, sorting, parallelism .... Particularly in the distributed executions fields, partially ordered sets have been used to modelize the message passing causality through logical clocks mechanism see Lamport [12], Fidge [6] and Mattern [13]. The two last time stamping algorithms allow to encode exactly the causality order "on–the–fly" that is during the execution. This "on–the–fly" notion is intrinsically required when dealing with distributed execution debugging (see for example Diehl et al. [5]) and leads to consider on–line algorithm on orders.

Among classes, interval orders constitute important and well–studied structures for both theoretical and practical reasons, see Golumbic [9], Fishburn [7] and Möhring [14]. Among applications interval orders have been recently used in the distributed programming debugging community. Working on concurrency measures, Habib et al. defined in [10] a distributed execution to be admissible under temporal dependency as being any interval order extension of the causality order. Working on models of concurrency, Janicki and Koutny in [11] defined observations of distributed execution to be particular interval order extensions of the causality order, arguing that maximal sets of concurrent (i.e. causally independent) events are observed and that the observer time scale is linearly ordered. Interval orders have also been used, in order to improve the "on–the–fly" encoding of the causality order, by Diehl and Jard in [4] where they give an on–line algorithm providing a constant encoding of the causality order under the hypothesis that the causality order yielded by a distributed execution running under a Remote Procedure Call message–passing is an interval order.

Recognition algorithms for interval orders have been proposed by Papadimitriou and Yannakakis [15], Möhring [14], Baldy and Morvan [2] and Garbe [8]. The two first algorithms have $\mathcal{O}(n+m)$ time and $\mathcal{O}(n^2)$ space complexities, the two last algorithms have $\mathcal{O}(n+m)$ time and space complexities, where n and m are respectively the number of elements and arcs of the order given in input. But all these algorithms are intrinsically off–line, that is they suppose the whole order known in advance. In this paper we present two on–line algorithms for recognizing interval orders under different hypothesis concerning the nature of the input data.

First of all let us describe our model of computation. All complexity results are calculated using the RAM model with uniform cost criterion see Aho, Hopcroft and Ullman [1]. On the other hand our on–line paradigm is the following: given a poset $P = (X, <_P)$ a new element $x \notin X$ and two possibly empty subsets of X, say $A(x)$ and $B(x)$, we have to consider the directed graph $G' = (X', E')$ where $X' = X \cup \{x\}$, the subgraph $G = (X, E)$ of the transitive closure of G' spanned by X is isomorphic to P and $E' = E \cup \{(y, x),\ y \in A(x)\} \cup \{(x, z),\ z \in B(x)\}$. Each subset is actually either made of direct access to the elements it represents or made of codings of the elements it represents. In the first case each access to an element takes $\mathcal{O}(1)$ time and in the second case it takes $\mathcal{O}(log(|X|))$ time by structuring the elements in a height–balanced tree.

This paper is broken down as follows. In section 2 we suppose that x comes with two any subsets $D(x)$ and $U(x)$ of X, what we call *"the suborder hypothesis"*, because if G' is an order we only known that P is a suborder of G'. In this case and under the hypothesis that P is an interval order, we prove that recognizing that G' is an interval order such that $D(x)$, resp. $U(x)$, is exactly the downset, resp. the upset, of x in G' can be done in $\mathcal{O}(|D(x)| + |U(x)|)$ time and in space linear to the size of G'. In section 3 we suppose that x comes with any subset $LC(x)$ of X, what we call *"the linear extension hypothesis"*, because G' is necessarily acyclic and x is maximal in G'. In this case and under the hypothesis that P is an interval order, we prove that recognizing that the transitive closure of G' is an interval order can be done in $\mathcal{O}(\Delta^+(P) + |LC(x)|)$ time and space linear to the size of the transitive reduction of G' assuming that P is given by its transitive reduction and that $LC(x)$ is an antichain of P.

A partially ordered set $P = (X, <_P)$ (poset or order for short) is an irreflexive and transitive binary relation on X.

With an order P are naturally associated two directed graphs: its transitive closure $G_c = (X, E_c)$ and its transitive reduction $G_r = (X, E_r)$, where $(x, y) \in E_c$ iff $x <_P y$ while $(x, y) \in E_r$ iff $x <_P y$ and there is no z in X such that $x <_P z$ and $z <_P y$, $(x, y) \in E_r$ is also denoted by $x -\!\!\prec_P y$.

For $x \in X$, we define $D_P(x) = \{y \in X,\ y <_P x\}$ the downset of x, $LC_P(x) = \{y \in X,\ y -\!\!\prec_P x\}$ the lower cover set of x, $U_P(x) = \{y \in X,\ x <_P y\}$ the upset of x and $UC_P(x) = \{y \in X,\ x -\!\!\prec_P y\}$ the upper cover set of x. An element x is said maximal if $U_P(x) = \emptyset$.

An order $P' = (X', <_{P'})$ is a suborder of $P = (X, <_P)$ if $X' \subseteq X$ and $\forall\ x, y \in X'$ we have $(x <_{P'} y \iff x <_P y)$; P is also denoted $P|_{X'}$.

A linear extension of P is any total order on X, say $L = x_1 < x_2 < \cdots < x_n$, such that $(x_i <_P x_j) \Longrightarrow (x_i <_L x_j)$.

In this paper, all posets are finite and we set $n(P) = |X|$, $m_c(P) = |E_c|$ and $m_r(P) = |E_r|$. We denote by $d^+_P(x)$ the outdegree of x in G_r and by $\Delta^+(P)$ the maximal outdegree of an element in G_r.

An order $P = (X, <_P)$ is an *interval order* iff we can associate to X a collection $(I_x)_{x \in X}$ of intervals of the real line such that $(x <_P y) \iff (I_x$ *lies strictly on the left of* $I_y)$ (i.e. $\forall\ a \in I_x$, $\forall\ b \in I_y$, we have $a <_{I\!R} b$ where "$<_{I\!R}$" is the usual strict order on real numbers).

The order on $\{a, b, c, d\}$ such that $a < c$ and $b < d$ are the only comparabilities is called a $2 \oplus 2$.

In the paper we need the following theorem introducing some of the most known characterization of interval orders.

Theorem 1 *Let $P = (X, \leq_P)$ be an order, then the following properties are equivalent:*

(i) *P is an interval order.*

(ii) *P does not contain a suborder isomorphic to the $2 \oplus 2$.*

(iii) *There exists a linear extension $L_U = x_1 < x_2 < \cdots < x_n$ of P such that $U_P(x_n) \subseteq \cdots \subseteq U_P(x_1)$.*

(iv) *There exists a linear extension $L_D = y_1 < y_2 < \cdots < y_n$ of P such that $D_P(y_1) \subseteq \cdots \subseteq D_P(y_n)$.*

(v) *$\forall\ x, y \in X,\ (U_P(x) \subseteq U_P(y)) \Longleftrightarrow (|\ U_P(x)\ | \leq |\ U_P(y)\ |)$.*

(vi) *$\forall\ x, y \in X,\ (D_P(x) \subseteq D_P(y)) \Longleftrightarrow (|\ D_P(x)\ | \leq |\ D_P(y)\ |)$.*

The properties *(iii)* and *(iv)* allow to define the total quotient orders $U(P) = U_1 > \cdots > U_q$ and $D(P) = D_1 > \cdots > D_r$ with respect to the equivalence relations $(x \sim_U y) \Longleftrightarrow (U_P(x) = U_P(y))$ and $(x \sim_D y) \Longleftrightarrow (D_P(x) = D_P(y))$. For $U(P)$ the induced order is the dual of the successor sets inclusion order whereas for $D(P)$ the induced order is the predecessor sets inclusion order.

2 Recognition under Suborder Hypothesis

Let $P = (X, <_P)$ be an interval order, let $x \notin X$. Let $D(x)$ and $U(x)$ be two subsets of X. Let $G' = (X', E')$ be the directed graph on $X' = X \cup \{x\}$ such that $E' = \{(y, z) \in X^2,\ y <_P z\} \cup \{(y, x),\ y \in D(x)\} \cup \{(x, z),\ z \in U(x)\}$ and such that the subgraph, of the transitive closure of G', spanned by X is isomorphic to P. The problem is to check if G' is isomorphic to an interval order P' such that $D(x) = D_{P'}(x)$ and $U(x) = U_{P'}(x)$. Recall from Section 1 that $U(P) = U_1 > \cdots > U_q$ and $D(P) = D_1 > \cdots > D_r$.

Let $G_1 = (X', E_1)$ and $G_2 = (X', E_2)$ be the two directed graphs such that $X' = X \cup \{x\}$, $E_1 = \{(y, z) \in X^2,\ y <_P z\} \cup \{(y, x),\ y \in D(x)\}$ and $E_2 = \{(y, z) \in X^2,\ y <_P z\} \cup \{(x, z),\ z \in U(x)\}$.

Property 1 *If $D(x) \cap U(x) = \emptyset$ then G' is isomorphic to an interval order P' iff G_1 and G_2 are isomorphic to interval orders.*

Proof: Using an interval representation of P' we obtain an interval representation of G_1 by increasing the larger extremity of the interval associated to x up to the largest extremity of an interval in this representation. We use dually the same argument for G_2.
For the converse, if G' is not isomorphic to an order either there exists a directed cycle C in G' or a transitivity arc is missing in E'. In the first case since P is an order and $D(x) \cap U(x) = \emptyset$ then $C = (a, b, x, a)$ with $a <_P b$, $a \in U(x)$, $b \in D(x)$ which contradicts that G_1 and G_2 are orders. In the second case there exists $a \in D(x)$, $b \in U(x)$ with $a \|_P b$ which contradicts the suborder hypothesis. Since P is an interval order, if P' is not an interval order then any $2 \oplus 2$ of P' contains x and belongs necessarily to G_1 or G_2. □

The following property gives the main structure of our recognition algorithm.

Property 2 *If $D(x) \cap U(x) = \emptyset$ then G' is isomorphic to an interval order iff*

(a) *If $D(x) \neq \emptyset$ then*

(i) $\exists\ k_U \in [1, q],\ \{i \in [1, q],\ D(x) \cap U_i \neq \emptyset\} = [k_U, q]$

(ii) $\forall\ i,\ k_U < i \leq q,\ U_i \subseteq D(x)$

(b) *If $U(x) \neq \emptyset$ then*

(iii) $\exists\ k_D \in [1, r],\ \{i \in [1, r],\ U(x) \cap D_i \neq \emptyset\} = [1, k_D]$

(iv) $\forall\ i,\ 1 \leq i < k_D,\ D_i \subseteq U(x)$

Proof: From Property 1, it is sufficient to prove that G_1 and G_2 are isomorphic to interval orders. We are going to show that G_1 is isomorphic to interval order P_1 iff it checks condition (a). Let $z \in U_{k_U}$ such that $z <_{P_1} x$. Consider $i > k_U$ and $y \in U_i$, then $U_P(z) \subset U_P(y)$. Since P_1 is an interval order, we have $U_{P_1}(z) \subset U_{P_1}(y)$ and so $y <_{P_1} x$.
Conversely, set $U''_k = \{y \in U_{k_U},\ y <_{P_1} x\}$ and $U'_k = U_k - U''_k$. If $k = 1$ then when $U'_k = \emptyset$ we have $U(P_1) = \{x\} > U_1 > \cdots > U_q$ and when $U'_k \neq \emptyset$ we have $U(P_1) = \{x\} \cup U'_1 > U''_1 > U_2 > \cdots > U_q$. Otherwise $U(P_1) = \{x\} \cup U_1 > \cdots > U_{k_U - 1} > U'_k > U''_k > U_{k_U + 1} > \cdots > U_q$. □

Then the algorithm can be informally presented as follow:

Algorithm 1:

Case 1 : If $(D(x) = \emptyset)$ *and* $(U(x) = \emptyset)$ then
Update(P', $U(P')$, $D(P')$);
Stop with P' is an Interval order;

Case 2 : If $(U(x) = \emptyset)$ then
Find $k_{_U} = min\{i \in [1,q],\ D(x) \cap U_i \neq \emptyset\}$;
If $(\forall\ i,\ k_{_U} < i \leq q,\ U_i \subseteq D(x))$ then
Update(P', $U(P')$, $D(P')$, $\{y,\ y \in D(x)\}$);
Stop with P' is an Interval order;
else Stop with P' is not an Interval order;

Case 3 : If $(D(x) = \emptyset)$ then
Find $k_{_D} = max\{i \in [1,r],\ U(x) \cap D_i \neq \emptyset\}$;
If $(\forall\ i,\ 1 \leq i < k_{_D},\ D_i \subseteq U(x))$ then
Update(P', $U(P')$, $D(P')$, $\{y,\ y \in U(x)\}$);
Stop with P' is an Interval order;
else Stop with P' is not an Interval order;

Case 4 : *(* We are in the case* $(D(x) \neq \emptyset)$ *and* $(U(x) \neq \emptyset)$ **)*
If $((D(x) \cap U(x)) \neq \emptyset)$ then Stop with P' is not an Interval order;

Case 5 : *(* We are in the case* $(D(x) \neq \emptyset)$, $(U(x) \neq \emptyset)$ *and* $(D(x) \cap U(x) = \emptyset)$ **)*
Find $k_{_U} = min\{i \in [1,q],\ D(x) \cap U_i \neq \emptyset\}$;
Find $k_{_D} = max\{i \in [1,r],\ U(x) \cap D_i \neq \emptyset\}$;
If $(\forall\ i,\ k_{_U} < i \leq q,\ U_i \subseteq D(x))$ *and* $(\forall\ i,\ 1 \leq i < k_{_D},\ D_i \subseteq U(x))$ then
Update(P', $U(P')$, $D(P')$, $\{y,\ y \in D(x) \cup U(x)\}$);
Stop with P' is an Interval order;
else Stop with P' is not an Interval order;

In order to perform the updating steps of the algorithm and to estimate its time complexity we need the following properties.

Assume that $U(x) \neq \emptyset$, we set $u = min\{i \in [1,q],\ |U_{_P}(U_i)| \geq |U(x)|\}$ if it exists and q otherwise.

Property 3 *If G' is isomorphic to an interval order P', then*

(i) If $(|U_P(U_u)|>|U(x)|)$ we have $U_{P'}(U_{u-1}) \subset U_{P'}(x) = U(x) \subset U_{P'}(U_u)$

(ii) If $(|U_P(U_u)|=|U(x)|)$ we have $U_{P'}(U'_u) = U_{P'}(x) = U(x) \subset U_{P'}(U''_u)$ with $U''_u = U_u \cap D(x)$ and $U'_u = U_u - U''_u$

Proof: Let $X_1 = \{y \in X, y <_{P'} x\}$ and let $X_2 = X - X_1$. From the suborder hypothesis we deduce that $\forall\ v \in X_1$, $U_{P'}(v) = U_P(v) \cup \{x\}$ and $\forall\ w \in X_2$, $U_{P'}(w) = U_P(w)$ which implies the property. □

In order to establish the complexity of Algorithm 1, we need the following result coming from the successor set inclusion property of interval orders.

Property 4 *If G' is isomorphic to an interval order P', then $u \leq |U(x)| + 1$*

Using similar arguments, we get the dual properties on $D(P')$.

Assume that $D(x) \neq \emptyset$, we set $d = max\{i \in [1,r],\ |D_P(D_i)| \geq |D(x)|\}$ if it exists and 1 otherwise.

Property 5 *If G' is isomorphic to an interval order P', then*

(i) If $(|D_P(D_d)|>|D(x)|)$ we have $D_{P'}(D_{d+1}) \subset D_{P'}(x) = D(x) \subset D_{P'}(D_d)$

(ii) If $(|D_P(D_d)|=|D(x)|)$ we have $D_{P'}(D'_d) = D_{P'}(x) = D(x) \subset D_{P'}(D''_d)$ with $D''_d = D_d \cap U(x)$ and $D'_d = D_d - D''_d$

Property 6 *If G' is isomorphic to an interval order P', then $r - d \leq |D(x)|$*

We are now able to state our main theorem:

Theorem 2 *Algorithm 1 checks if G' is isomorphic to an interval order P' within $\mathcal{O}(|D(x)| + |U(x)|)$ time and $\mathcal{O}(n(P') + m_c(P'))$ space.*

Proof: Let us describe the data structures used by the algorithm.

- The sets $D(x)$ and $U(x)$ are given by the list of their elements.
- The order $U(P)$ is represented by a doubly linked list of the classes in the order $U_1, \cdots, U_q$. Each U_i is a record constituted of six fields, namely the two first fields are pointers for the double linking; the third field contains an integer initialized to the null value (used to find $k_{_U}$ and to check the inclusion property of the reached U_i's); the fourth field contains an integer representing the size of U_i; the fifth field is a pointer to the data structure associated to P (it is useful only for the head and the tail of the list) and the last one is a pointer to a doubly linked list of the elements of U_i (each element is a four fields record, the first one is a pointer to the element it represents, the two next fields are pointers for the double linking while the last one is a pointer to the class it belongs).
- The same data structures as above are used for $D(P)$.
- To each $y \in X$ is associated a five fields record namely: two integers respectively corresponding to the cardinality of its successor set and to the cardinality of its predecessor set; two pointers respectively pointing to the record associated to y in $D(P)$ and in $U(P)$ and a boolean field initialized to the False value (used for testing if $D(x) \cap U(x) = \emptyset$, this field is not necessary for this test but is helpful for clarity).
- To P is associated four pointers pointing respectively to the head and the tail of the list representing $U(P)$ and to the head and the tail of the list representing $D(P)$.

For $Case\ 1\ (D(x) = \emptyset)\ and\ (U(x) = \emptyset)$:
$U(P') = \{x\} \cup U_1 > \cdots > U_q$ and $D(P') = D_1 > \cdots > D_r \cup \{x\}$. Then updating $U(P')$ and $D(P')$ takes clearly $\mathcal{O}(1)$ time.

The data structures allow to find $k_{_U}$ (resp. $k_{_D}$) and check the inclusion property of the reached U_i's (resp. D_i's) in $\mathcal{O}(|D(x)|)$ (resp. $\mathcal{O}(|U(x)|)$) time. Indeed, take $U(P)$, it is sufficient to list $D(x)$ and during this listing to both increase the third field of the reached U_i's by one and keep the address of the reached U_i with a maximal $|U_{_P}(U_i)|$. The selected U_i is then $U_{k_{_U}}$.

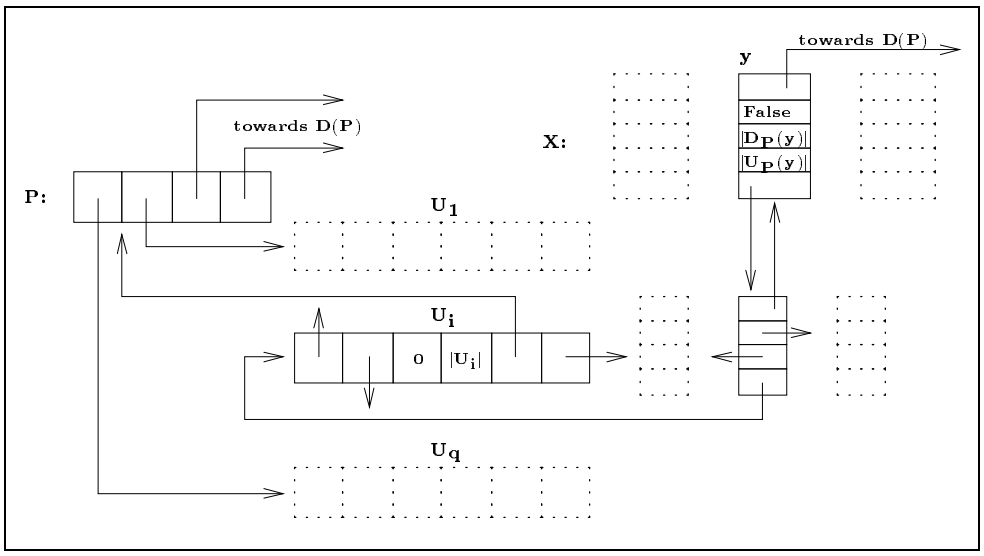

Figure 1: Data structures used by Algorithm 1

In order to verify the inclusion property we only have to check that starting from the tail of $U(P)$, all the U_i's reached until U_{k_U}, U_{k_U} excluded, have the same value in their third and fourth field. The same holds dually for finding k_D in $\mathcal{O}(|U(x)|)$.

Now, assuming that G' is isomorphic to an interval order P', we have to update the data structures. In order to do that, we have to define $U(P')$ and $D(P')$.

For *Case* 2 "$(D(x) \neq \emptyset)$ *and* $(U(x) = \emptyset)$":
If $k_U = 1$ and $U_1 \subseteq D(x)$ then $U(P') = \{x\} > U_1 > \cdots > U_q$ and $D(P') = \{x\} > D_1 > \cdots > D_r$. Otherwise, let $d = \max_{i \in [1,r]} \{i,\ |D_P(D_i)| \geq |D(x)|\}$. If $|D_P(D_d)| > |D(x)|$ then $D(P') = D_1 > \cdots > D_d > \{x\} > D_{d+1} > \cdots > D_r$ else $D(P') = D_1 > \cdots > (D_d \cup \{x\}) > \cdots D_r$. If $k_U = 1$ (we have $U_1 \not\subseteq D(x)$) then let $U_1'' = U_1 \cap D(x)$ and let $U_1' = U_1 - U_1''$, so $U(P') = (\{x\} \cup U_1') > U_1'' > U_2 > \cdots > U_q$, else (we have $k_U \neq 1$) let $U''_{k_U} = U_{k_U} \cap D(x)$ and let $U'_{k_U} = U_{k_U} - U''_{k_U}$, so $U(P') = (\{x\} \cup U_1) > \cdots > U_{k_U - 1} > U'_{k_U} > U''_{k_U} > U_{k_U + 1} > \cdots > U_q$.

For $Case\ 3$ "$(D(x) = \emptyset)\ and\ (U(x) \neq \emptyset)$":
If $k_{_D} = r$ and $D_r \subseteq U(x)$ then $D(P') = D_1 > \cdots > D_r > \{x\}$ and $U(P') = U_1 > \cdots > U_q > \{x\}$. Otherwise, let $u = \min\limits_{i\in[1,q]} \{i,\ |U_{_P}(U_i)|\geq|U(x)|\}$. If $|U_{_P}(U_u)|>|U(x)|$ then $U(P') = U_1 > \cdots > U_{u-1} > \{x\} > U_u > \cdots > U_q$ else $U(P') = U_1 > \cdots > (U_u \cup \{x\}) > \cdots U_q$. If $k_{_D} = r$ (we have $D_r \not\subseteq U(x)$) then let $D''_r = D_r \cap U(x)$ and $D'_r = D_r - D''_r$, so $D(P') = D_1 > \cdots > D_{r-1} > D''_r > (D'_r \cup \{x\})$ else (we have $k_{_D} \neq r$) let $D''_{k_{_D}} = D_{k_{_D}} \cap U(x)$ and $D'_{k_{_D}} = D_{k_{_D}} - D''_{k_{_D}}$, so $D(P') = D_1 > \cdots > D_{k_{_D}-1} > D''_{k_{_D}} > D'_{k_{_D}} > D_{k_{_D}+1} > \cdots > (D_r \cup \{x\})$.

For $Case\ 4$ "$((D(x) \cap U(x)) \neq \emptyset)$":
We can check if $D(x) \cap U(x) = \emptyset$ in $\mathcal{O}(|D(x)| + |U(x)|)$ time by listing both $D(x)$ and $U(x)$ and using for boolean field of the elements of X.

For $Case\ 5$ "$(D(x) \neq \emptyset)$, $(U(x) \neq \emptyset)\ and\ ((D(x) \cap U(x)) = \emptyset)$":
Let $U''_{k_U} = U_{k_U} \cap D(x)$ and let $U'_{k_U} = U_{k_U} - U''_{k_U}$. Let $u = \min\limits_{i\in[1,q]} \{i,\ | U_{_P}(U_i)|\geq|U(x)|\}$ if $|U_{_P}(U_u)|>|U(x)|$ then $U(P') = U_1 > \cdots > U_{u-1} > \{x\} > U_u > \cdots > U_{k_U-1} > U'_{k_U} > U''_{k_U} > U'_{k_U+1} > \cdots > U_q$. Otherwise (we have $|U_{_P}(U_u)|=|U(x)|$), if $u \neq k_U$ then $U(P') = U_1 > \cdots > U_u \cup \{x\} > \cdots > U_{k_U-1} > U'_{k_U} > U''_{k_U} > U_{k_U+1} > \cdots > U_q$ else $(u = k_U)$ $U(P') = U_1 > \cdots > U_{k_U-1} > U'_{k_U} \cup \{x\} > U''_{k_U} > U_{k_U+1} > \cdots > U_q$ (remark that U'_{k_U} is possibly the empty set).
Let $D''_{k_D} = D_{k_D} \cap U(x)$ and let $D'_{k_D} = D_{k_D} - D''_{k_D}$. Let $d = \max\limits_{i\in[1,r]} \{i,\ | D_{_P}(D_i)|\geq|D(x)|\}$ if $|D_{_P}(D_d)|>|D(x)|$ then $D(P') = D_1 > \cdots > D_{k_D-1} > D''_{k_D} > D'_{k_D} > D_{k_D+1} > \cdots > D_d > \{x\} > D_{d+1} > \cdots > D_r$. Otherwise (we have $|D_{_P}(D_d)|=|D(x)|$), if $d \neq k_D$ then $D(P') = D_1 > \cdots > D_{k_D-1} > D''_{k_D} > D'_{k_D} > D_{k_D+1} > \cdots > D_d \cup \{x\} > \cdots > D_r$ else $(d = k_D)$ $D(P') = D_1 > \cdots > D_{k_D-1} > D''_{k_D} > D'_{k_D} \cup \{x\} > D_{k_D+1} > \cdots > D_r$ (remark that D'_{k_D} is possibly the empty set).

The correctness of $U(P')$ and $D(P')$ is achieved through Property 3 and Property 5.

Since we have already determined U_{k_U} and D_{k_D} we have to find U_u (resp. D_d) where $u = min\{i \in [1,q],\ |U_P(U_i)| \geq |U(x)|\}$ (resp. $d = max\{i \in [1,r],\ |D_P(D_i)| \geq |D(x)|\}$). With Property 4, Property 6 and the data structures this can be done in $\mathcal{O}(|D(x)| + |U(x)|)$ time. Indeed, for u, it can be achieved in $\mathcal{O}(|U(x)|$ time by listing $U_1, U_2, \cdots$ until to find U_u by comparing the number of successors in P of any element in the current U_i to $|U(x)|$. For d, using the same procedure starting with D_r, we get d in $\mathcal{O}(|D(x)|$.

Now depending on which case we are, it remains to add an element to a class, create a new class or to split a class. Carrying out these operations and their corresponding updates take $\mathcal{O}(1)$ time in the two first cases and $\mathcal{O}(|U(x)|)$ or $\mathcal{O}(|D(x)|)$ time for the last one. Indeed, the only tricky point is the splitting U_{k_U} in U'_{k_U} and U''_{k_U} which is achieved by creating a new class just after U_{k_U} in $U(P)$, listing $D(x)$ and moving all its elements in U_{k_U} to the new class U''_{k_U}. The resulting U_{k_U} is now the class U'_{k_U}.

As last updates, for all y belonging to $D(x)$ (resp. $U(x)$), we have to add one to the integer corresponding to $|U_P(y)|$ (resp. $|D_P(y)|$). We have to restore to the null value the third field of all the reached U_i's and D_i's and if necessary to restore to the False value the boolean field of the elements belonging to $D(x)$ or $U(x)$. The update of the head and tail pointers is assumed to be done when necessary in the corresponding steps. The update of the four record fields of x are also done during the accurate steps. All of this can be clearly done in $\mathcal{O}(|D(x)| + |U(x)|)$ time.

The space complexity is clearly in $\mathcal{O}(n(P') + m_c(P'))$. □

Given any order $P = (X, <_P)$ the previous result can be naturally used to check if P is an interval order by considering successively $P_1, \cdots, P_n = P$ where P_i is the suborder of P induced by $\{x_1, \cdots, x_i\}$ and $(x_1, \cdots, x_n)$ is any ordering of X. Then we have:

Corollary 1 *Recognizing if an order P, given by its transitive closure, is an interval order, can be done in $\mathcal{O}(n(P) + m_c(P))$ time and space.*

3 Recognition under Linear Extension Hypothesis

In Section 2 our on–line algorithm recognizes interval order in time and space linear in the size of its transitive closure. Then it is natural to ask if the same result holds for the transitive reduction. But our approach fails under the suborder hypothesis since updating the predecessor and the successor classes for the incoming vertex requires a time linear in the size of the transitive closure. Nevertheless under the linear extension hypothesis we obtain an almost optimal result.
Let $P = (X, \leq_P)$ be an interval order, let $x \notin X$. Let $P' = (X', \leq_{P'})$ be an order where $X' = X \cup \{x\}$, $P'|_X = P$ and $x \in Max(P')$. Let $LC(x)$ be a subset of X which is the immediate predecessor set of x in P', $LC(x) = \{y \in X,\ y -<_{P'} x\}$. The problem is to check if $P' = (X', \leq_{P'})$ is also an interval order. Recall from Section 1 that $U(P) = U_1 > \cdots > U_q$. Our next algorithm is based on the following property.

Property 7 *Assume that $LC(x) \neq \emptyset$, then P' is an interval order iff*

(i) $\exists\ k, l \in [1, q],\ \{i \in [1, q],\ LC(x) \cap U_i \neq \emptyset\} = [k, l]$

(ii) $\forall\ i,\ k < i \leq l,\ U_i \subseteq LC(x)$

(iii) $\forall\ z \in \bigcup_{j=l+1}^{q} U_j,\ \exists\ y \in LC(x)$ *such that* $z <_P y$

Proof: Let $z \in U_k$ with $z \in LC(x)$. Assume that there exists $y \in U_i$ with $i \in]k, l]$ such that $y \notin LC(x)$. Since P is an interval order we have $U_P(z) \subset U_P(y)$ and since P' is an interval order we have $x \in U_{P'}(y)$. This implies that there exists $j \in [k, i[$ such that there exists $v \in U_j \cap LC(x)$ and $v \in U_{P'}(y)$. Consider $t \in U_l \cap LC(x)$ since P' is an interval order we have $U_{P'}(y) \subseteq U_{P'}(t)$ and so $v \in U_{P'}(t)$ which is a contradiction. The condition (iii) is immediate.
Conversely, set $U''_k = \{y \in U_k,\ y <_{P'} x\}$ and $U'_k = U_k - U''_k$. If $k = 1$ then when $U'_k = \emptyset$ we have $U(P') = \{x\} > U_1 > \cdots > U_q$ and when $U'_k \neq \emptyset$ we have $U(P') = \{x\} \cup U'_1 > U''_1 > U_2 > \cdots > U_q$. Otherwise $U(P') = \{x\} \cup U_1 > \cdots > U'_k > U''_k > \cdots > U_q$. □

Then the algorithm can be informally presented as follow:

Algorithm 2:

Case 1 : If $(LC(x) = \emptyset)$ then

Update(P', $U(P')$);
Stop with P' is an Interval order;

Case 2 : If $(LC(x) \neq \emptyset)$ then

Find $k = min\{i \in [1,q],\ LC(x) \cap U_i \neq \emptyset\}$;
Find $l = max\{i \in [1,q],\ LC(x) \cap U_i \neq \emptyset\}$;
If $(\forall\ i,\ k < i \leq l,\ U_i \subseteq LC(x))$ *and* $(UC_P(U_{l+1}) \cap LC(x) \neq \emptyset)$ then
Update(P', $U(P')$, $\{UC(y),\ y \in LC(x)\}$);
Stop with P' is an Interval order;
else Stop with P' is not an Interval order;

Theorem 3 *Algorithm 2 checks if P' is an interval order within $\mathcal{O}(\Delta^+(P) + |LC(x)|)$ time and $\mathcal{O}(n(P') + m_r(P'))$ space.*

Proof: Let us describe the data structures used by the algorithm.

- The set $LC(x)$ is given by the list of its elements.
- The order $U(P)$ is represented by a doubly linked list of the classes in the order $U_1, \cdots, U_q$. Each U_i is a record constituted of five fields, namely the two first fields are pointers for the double linking; the third field contains an integer initialized to the null value (used to find k and l and to check the interval property of the reached U_i's); the fourth field contains an integer representing the size of U_i; and the last one is a pointer to a doubly linked list of the elements of U_i (each elements is a four fields record, the first one is a pointer to the element it represent, the two next fields are pointers for the double linking while the last one is a pointer to the class it belongs).
- To each $y \in X$ is associated a two fields record namely, a pointer to the record associated to y in $U(P)$ and a pointer to the list of it's immediate successors (each element is a two fields record the first one is a pointer for the linking while the last one is a pointer to the element it represents).

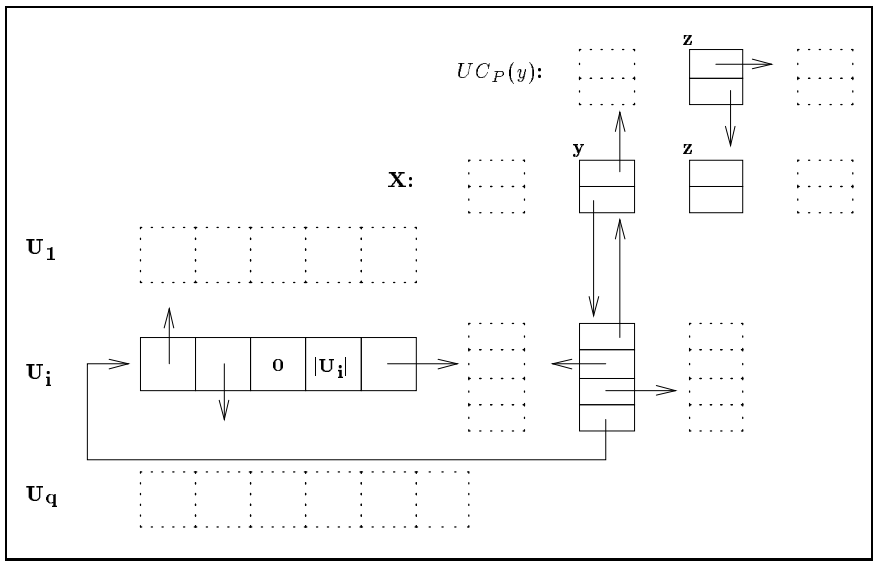

Figure 2: Data structures used by Algorithm 2

With this data structure the algorithm runs as follows:

For $Case\ 1$ "$LC(x) = \emptyset$":
We have $U(P') = \{x\} \cup U_1 > \cdots > U_q$, then updating P' and the immediate successor sets takes clearly $\mathcal{O}(1)$ time.

For $Case\ 2$ "$LC(x) \neq \emptyset$":
Computing k, l and checking $(\forall\ i,\ k < i \leq l,\ U_i \subseteq LC(x))$ can be done in $\mathcal{O}(|LC(x)|)$ time during the same step. Indeed we list $LC(x)$ and for every y in $LC(x)$ we increase by one the third field of its class. Take any y in $LC(x)$, let U_i be its class. Consider the largest interval of classes containing U_i such that their third field has a not null value, say $U_{i_1} < \cdots < U_{i_2}$. If the sum of the third field of the classes in this interval is equal to $|LC(x)|$ then $k = i_1$ and $l = i_2$ otherwise conditions (i) and (ii) of Property 7 are not satisfied.
It remains to verify $(UC_P(U_{l+1}) \cap LC(x) \neq \emptyset)$. For that purpose we have to split U_k in $U_k'' = U_k \cap LC(x)$ and $U_k' = U_k - U_k''$ and we have to update with the correct values the corresponding records, particularly the third fields

contains the null value for U'_k and a non null value for U''_k. If $U_{l+1} \neq \emptyset$, for $z \in U_{l+1}$ we have to check that at least one of its immediate successor belongs to a class with a non null value in its third field. This step is achieved in $\mathcal{O}(d^+(z))$ time.
For updating $U(P')$ we have just to restore with the null value all the third field of the classes corresponding to $LC(x)$ and to construct the immediate successor sets of any y in $LC(x)$.

The space complexity is clearly in $\mathcal{O}(n(P') + m_r(P'))$. □

Given any order $P = (X, <_P)$ the previous result can be naturally used to check if P is an interval order by considering successively $P_1, \cdots, P_n = P$ where P_i is the suborder of P induced by $\{x_1, \cdots, x_i\}$ and $(x_1, \cdots, x_n)$ is a linear extension P (computing a linear extension of an order given by its transitive reduction is linear with respect to its number of vertices and arcs). Then we have:

Corollary 2 *Recognizing if an order P, given by its transitive reduction, is an interval order, can be done in $\mathcal{O}(n(P) + n(P)\Delta^+(P))$ time and $\mathcal{O}(n(P) + m_r(P))$ space.*

4 Conclusion

In this paper we have presented two on–line algorithms to recognize an interval order. These algorithms can be naturally adapted to work incrementally if the whole order is known in advance. The first one recognizes the transitive closure of the order in optimal time and space under what we call "the suborder hypothesis". The second one recognizes the transitive reduction of the order in optimal space and almost optimal time under what we call "the linear extension hypothesis". As far as we know, these on–line algorithms are the fastest under the hypotheses we made.

However some open questions remain. Is it possible to obtain a linear space and time on–line algorithm to recognize the transitive reduction of an interval order under the linear extension hypothesis? or, but its seems more

difficult, under the suborder hypothesis? Assuming that the input is an acyclic digraph, the linear time and space algorithm given in [2], recognizes if its transitive closure is an interval order. Does there exist a linear time and space on–line algorithm which solves the same problem under the suborder or the linear extension hypothesis?

Acknowledgments

We would like to thank C. Jard and G.V. Jourdan for helpful comments on earlier versions. This work has received a financial support from the national project "Trace" of the French research ministry and from the French–Israeli research cooperation.

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The design and Analysis of Computer Algorithms*, Addison–Wesley Series in Computer Science and Information Processing, 1974.

[2] P. Baldy, M. Morvan, *A Linear Time and Space Algorithm to Recognize Interval Orders*, R.R. N° 92-015, LIRMM Montpellier, 1992, to appear in Discrete Applied Mathematics.

[3] V. Bouchitté, M. Habib, *The Calculation of Invariants of Ordered Sets*, I. Rival (ed.), Algorithms and Order, 231–279, NATO Series C - Vol 255, Kluwer Academic Publishers, 1989.

[4] C. Diehl, C. Jard, *Interval Approximations of Message Causality in Distributed Executions*, STACS'92: Theoretical Aspects of Computer Science, in Lecture Notes in Computer Science N^o 577, Springer–Verlag 1992, pp. 363–376.

[5] C. Diehl, C. Jard, J.X. Rampon, *Reachability Analysis on Distributed Executions*, TAPSOFT'93: Theory and Practice of Software Development, in Lecture Notes in Computer Science N^o 668, Springer–Verlag 1993, pp. 629–643.

[6] J. Fidge, *Timestamps in Message Passing Systems that Preserve the Partial Ordering*, In Proc. 11^{th} Australian Computer Science Conference, 55–66, February 1988.

[7] P.C. Fishburn, *Interval Orders and Interval Graphs*, Wiley, 1985.

[8] R. Garbe, *A Linear Time Algorithm for Recognizing Interval Orders*, Memorandum n^o 1046, Dept. of Math., University of Twente, 1992.

[9] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, New York, 1980.

[10] M. Habib, M. Morvan, J.X. Rampon, *Remarks on some Concurrency Measures*, Graph–Theoretic Concepts in Computer Science, in Lecture Notes in Computer Science N^o 484, Springer–Verlag 1990, pp. 221–238.

[11] R. Janicki, M. Koutny, *Structure of Concurrency*, Theoretical Computer Science 112 (1993) 5–52.

[12] L. Lamport, *Time, Clocks and the Ordering of Events in a Distributed System*, Communications of the ACM, 21(7): 558–565, July 1978.

[13] F. Mattern, *Virtual Time and Global States of Distributed Systems*, In Cosnard, Quinton, Raynal and Robert, editor, Proc. Int. Workshop on Parallel and Distributed Algorithms, North–Holland, 1989.

[14] R.H. Möhring, *Computationally Tractable Classes of Ordered Sets*, I. Rival (ed.), Algorithms and Order, 105–193, NATO Series C - Vol 255, Kluwer Academic Publishers, 1989.

[15] C. H. Papadimitriou, M. Yannakakis, *Scheduling Interval–Ordered Task*, SIAM Journal of Computing $Vol.$ 8, N^o 3, August 1979, 405–409.

[16] M. Pouzet, D. Richard (ed) *Orders: Description and Roles*, Annals of Discrete Mathematics (23), North-Holland, 1984.

[17] I. Rival (ed.), *Ordered Sets*, NATO Series C - Vol 83, D.Reidel Publishing Company, 1982.

[18] I. Rival (ed.), *Graphs and Order*, NATO Series C - Vol 147, D.Reidel Publishing Company, 1985.

[19] I. Rival (ed.), *Combinatorics and Ordered Sets*, Proc. of the AMS–IMS–SIAM joint Summer Research Conference, Contemporary Mathematics, Vol 57, AMS, 1986.

[20] I. Rival (ed.), *Algorithms and Order*, NATO Series C - Vol 255, Kluwer Academic Publishers, 1989.

Unité de recherche INRIA Lorraine, Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399