Discrete Mathematics 111 (1993) 289–303
North-Holland

On the calculation of transitive reduction–closure of orders

M. Habib, M. Morvan and J.-X. Rampon

LIRMM (CNRS & Université de Montpellier II) 860, rue de Saint-Priest, F 34090, Montpellier, France

Received 22 July 1991

Abstract

Habib, H., M. Morvan and J.-X. Rampon, On the calculation of transitive reduction–closure of orders, Discrete Mathematics 111 (1993) 289–303.

Computations of transitive closure and reduction of directed acyclic graphs are mainly considered in this paper. Classes of directed acyclic graphs for which such problems can be solved in linear time complexity (in accordance with the number of arcs) are proposed, namely: generalized N-free graphs, graphs such that the external or internal degree of any vertex is bounded in the transitive reduction, and bounded decomposition width orders, strongly W-free orders. For this purpose, we study the worst-case complexity of a nice algorithm. This algorithm is due to Goralcikova and Koubek (1979) and already studied by Mehlhorn (1984) and Simon (1988). Furthermore, two new classes of orders (generalized N-free and strongly W-free) which seem to have nice computational properties, are proposed and studied.

1. Introduction

Transitive closure and reduction problems have received some attention in the 1970s [2, 6, 7, 13, 18]. It was shown that an $O(n^a)$, with $a \geqslant 2$, algorithm to compute the transitive closure of an n-node directed graph (digraph), exists if and only if the product of two boolean $n \times n$ matrices can be computed in $O(n^a)$. Aho et al. [1] showed a similar result between the transitive closure and reduction problems.

The best known algorithm for matrix multiplication works in $O(n^{2.376})$ and is by Coppersmith and Winograd [5]. Therefore, the search of 'good' (of linear or even quadratic time complexity) algorithms for transitive closure or reduction, working on some particular classes of graphs and orders is still an interesting and fundamental problem, since any progress in this direction would imply better algorithms for matrix multiplication (see, for example, [14]).

Correspondence to: M. Habib, LIRMM, 860, rue de Saint Priest, 34100 Montpellier, France.

0012-365X/93/$06.00

The particular case of acyclic digraphs is, in fact, the kernel of both problems (transitive reduction and closure) since, for the transitive closure, one can first search in linear time for the strongly connected components and reduce the problem to the same problem on quotient acyclic digraph. For the transitive reduction, one can use Simon's [15] algorithm to obtain in linear time a transitively reduced graph for each strongly connected component of the original digraph.

Due to the many applications of transitive closure and reduction algorithms, Mehlhorn [10] and Simon [15] studied carefully a nice general-purpose algorithm for transitive closure of acyclic graphs, due to Goralcikova and Koubek [8].

In particular, for orders, many algorithms assume that the input is given either in its transitively closed or transitively reduced form. So, the precomputation time may be more important than the algorithm itself. Thus, it is very interesting to know for which classes of orders efficient algorithms are available. Bordat [4] showed transitive reduction algorithms for gradued and semi-modular acyclic graphs. In this direction, some theoretical progress was recently obtained by Ma and Spinrad [9] for particular classes of orders (namely two-dimensional orders, N-free orders and interval orders) and we present here some partial results in connections with both points of view.

Notations. Let $G=(X, E)$ be an acyclic digraph. We denote by $G_{tr}=(X, E_{tr})$ and $G^{tc}=(X, E^{tc})$, respectively, the transitive reduction and closure of G. Moreover, throughout this paper, $n=|X|$, $m=|E|$, $m_{tr}=|E_{tr}|$ and $m^{tc}=|E^{tc}|$.

The problem we are concerned with is the computation of G_{tr} and G^{tc}, respectively, in $O(n+m)$ or $O(n+m^{tc})$, for an acyclic digraph G. Let us note that all over the text, we call chain a directed path (because of the order terminology), and that, unless otherwise stated, all graphs are acyclic.

In all the figures, the implicit orientation of the arcs is bottom-up.

Our purpose is twofold, first we present some algorithms based on the computation of the rank function, by generalizing some of the results of Ma and Spinrad obtained for N-free orders to a wider class of graphs. In Section 3 we study the structures of the worst cases of the Goralcikova and Koubek top-down transitive closure and reduction algorithm.

2. Algorithms based on the rank function

The rank function can be useful to compute the transitive reduction in linear time (in accordance with the number of arcs). Such a way has been already used in [17] to compute the transitive reduction of series–parallel acyclic digraphs. In this section we introduce a class of graphs (containing the class of N-free orders and, therefore, the series–parallel orders) for which it is possible to compute the transitive reduction in linear time using only the rank function.

Definition 2.1. Let ρ^+ and ρ^- be, respectively, the rank and antirank functions associated with G (i.e. $\rho^+(x)$ is the maximal length of a chain ending on x and $\rho^-(x)$ is the maximal length of a chain starting on x). Let x be a vertex of G; we define

$$\rho^+\text{-inf}(x)=\left\{ y\in \text{Succ}_G(x) \text{ s.t. } \rho^+(y)=\min_{z\in \text{Succ}_G(x)} \rho^+(z)\right\},$$

$$\rho^-\text{-sup}(x)=\left\{ y\in \text{Succ}_G(x) \text{ s.t. } \rho^-(y)=\max_{z\in \text{Succ}_G(x)} \rho^-(z)\right\}.$$

Let FG_1 and FG_2 be the substructures of Fig. 1 (FG_1 and FG_2 are defined relatively to a graph G since there are constraints on the rank and antirank functions).

The following theorem gives a characterization in terms of forbidden configurations of a class of graphs for which transitive reduction can be computed using rank and antirank functions.

Theorem 2.2. *The following statements are equivalent:*

(i) $\forall x\in X$, $\text{Succ}_{G_{\text{tr}}}(x)=\rho^+\text{-inf}(x)\cup\rho^-\text{-sup}(x)$,

(ii) G_{tr} *does not contain any substructure isomorphic to* FG_1 *or* FG_2.

Proof. (i) $\Rightarrow$ (ii): Assume that G contains one of the two forbidden substructures with the conditions on the ranks; obviously, the vertex y belongs to $\text{Succ}_{G_{\text{tr}}}(x)$ but not to $\rho^+\text{-inf}(x)\cup\rho^-\text{-sup}(x)$.

(ii) $\Rightarrow$ (i): It is obvious that $(\rho^+\text{-inf}(x)\cup\rho^-\text{-sup}(x))\subseteq \text{Succ}_{G_{\text{tr}}}(x)$. Assume that there exists $y\in\text{Succ}_{G_{\text{tr}}}(x)$ such that $y\notin(\rho^+\text{-inf}(x)\cup\rho^-\text{-sup}(x))$. Let us now show that G contains one of the two forbidden substructures. As $y\notin\rho^+\text{-inf}(x)$, $\exists z, t\in X$ such that (t, z, y) is a chain of G_{tr} and $\rho^+(t)\geqslant\rho^+(x)$, $\rho^+(z)=\rho^+(t)+1=\rho^+(y)-1$, and $\exists y'\in\text{Succ}_{G_{\text{tr}}}(x)$ such that $\rho^+(y')<\rho^+(y)$. Since $y\notin\rho^-\text{-sup}(x)$, $\exists y''$, $z'\in X$ such that (x, y'', z') is a chain of G_{tr} and $\rho^-(z')\geqslant\rho^-(y)$. Now two cases are possible:

– $y'=y''$: $(t, x)\notin E_G$ since $\rho^+(t)\geqslant\rho^+(x)$; $(z, y')\notin E_G$ since $\rho^+(y')<\rho^+(y)$ and $\rho^+(z)=\rho^+(y)-1$; $(y, z')\notin E_G$ since $\rho^-(z')\geqslant\rho^-(y)$. These remarks give us the second forbidden substructure and the conditions on the ranks.

– $y'\neq y''$: For the same reasons (t, x), (z, y'), $(y, z')\notin E_G$; we can note that (z, y'') is allowed for E_G. These remarks give us the first forbidden substructure and the conditions on the ranks. □

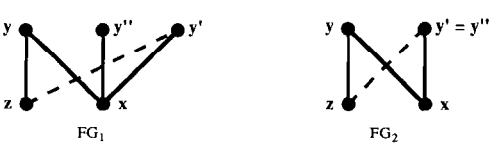

Fig. 1. The substructures FG_1 and FG_2 in which $\rho^+(y)>\rho^+(y')$ and $\rho^-(y'')>\rho^-(y)$. Dashed edges belong to the uncomparability graph of FG_i^{tc}.

Let us call *generalized N-free graph* a graph G satisfying the above conditions of Theorem 2.2.

Corollary 2.3. *Let $G=(X, E_G)$ be a generalized N-free graph, then the computation of its transitive reduction can be done in $O(n+m)$.*

Proof. The result comes immediately from the fact that rank and antirank functions can be computed in $O(n+m)$ time by a breadth-first search algorithm. Furthermore, using condition (i) of Theorem 2.2, for generalized N-free graphs, transitive reduction can be totally deduced from rank and antirank. □

Ma and Spinrad [9] proved that the computation of the transitive reduction of an N-free graph (i.e. an acyclic graph which does not contain any subgraph isomorphic to an N in its transitive reduction, as an order such a structure is called N-free order) can be done in $O(n+m)$. Corollary 2.3 also yields this result.

Corollary 2.4. *Let $G=(X, E_G)$ (with $|X|=n$ and $|E_G|=m$) be an acyclic digraph such that G_{tr} is the transitive reduction of an N-free order. Then the computation of the transitive reduction of G can be done in $O(n+m)$.*

Proof. Since the set of vertices $\{z, y, x, y'\}$ of the two graphs of Fig. 1 is the forbidden structure of a N-free order, the class of N-free orders is contained in the class of generalized N-free graphs, and the result follows. □

Generalized N-free is a strict generalization of N-free graphs, since it contains all bipartite graphs. Furthermore, it must be mentioned that a linear-time recognition algorithm for N-free graphs is still an open problem, when the input is the graph G itself (not necessarily the transitive reduction). A similar question may be asked for generalized N-free graphs.

Furthermore, orders with a unique maximal element w such that for any element x, all paths from x to w have the same length, can be easily proved to be generalized N-free. (Similarly for the dual condition on a unique minimal element.) In particular, this applies to lattices having Jordan–Dedekind condition.

3. A top-down algorithm for transitive closure and reduction

In this section we study in an accurate way a natural algorithm computing the transitive closure and reduction of acyclic digraphs.

3.1. The transitive closure and reduction algorithm

Definition 3.1. We define an *L-ordered representation* of an acyclic digraph $G=(X, E)$ according to a linear extension L of G, as a successors list representation of G such that for each vertex the successors are sorted according to L.

We can remark that the computation of an L-ordered representation of an acyclic digraph is no more expensive than the computation of the linear extension L.

Proposition 3.2. *An L-ordered representation of an acyclic graph $G=(X, E)$ can be computed in $O(|X|+|E|)$ time.*

Proof. Let us suppose that the graph is given by its predecessors list. We just have to build the successors list of each vertex in the following way (that is obviously in linear time):

(i) Take each a_i in its order of appearance in L.

(ii) For each $x \in \mathrm{Pred}_G(a_i)$, add a_i at the end of the successors list of x.

Since L is a linear extension, it is easy to deduce from step (i) that if $a_i, a_j \in \mathrm{Succ}_G(x)$ then a_i is before a_j in the successors list of x if and only if $i<j$. The result follows. □

By this proposition, we can now use this data structure in the following transitive closure and reduction algorithm. The following top-down algorithm is a slight modification of the one first proposed by Goralcikova and Koubek [8], and studied by Mehlhorn [10] and Simon [15], and uses the graph structure to avoid as much as possible dummy calculations. It builds up from sinks to sources the successors sets in the transitive closure. The tricky point is to transmit successors sets only alongside the arcs of the transitive reduction. In what follows we somehow reconsider the analysis of this algorithm, by trying to characterize the worst cases.

Top-down algorithm. Let $G=(X, E)$ be an acyclic digraph, $G_{\mathrm{tr}}=(X, E_{\mathrm{tr}})$ its transitive reduction and $G^{\mathrm{tc}}=(X, E^{\mathrm{tc}})$ its transitive closure. Let $L=a_1, \ldots, a_n$ be a linear extension of G.

Begin
For all $x \in X$ **do**
  begin
  $s_{G_{\mathrm{tr}}}(x) := \emptyset$;
  $s_{G^{\mathrm{tc}}}(x) := \emptyset$;
  end;
{Take each vertex in the inverse order of appearance in L and compute its set of successors in G^{tc}}
For $i := n$ **down to** 1 **do**
  begin
  $S := \mathrm{Succ}_G(a_i)$;
  While $S \neq \emptyset$ **do**
    begin
    Take $x \in \min(L/S)$;
    If x is not marked **then**
    {(a_i, x) is not a transitivity edge}

```
        begin
        s_{G_tr}(a_i):=s_{G_tr}(a_i)∪{x};
        s_{G^tc}(a_i):=s_{G^tc}(a_i)∪{x};
        For all y∈s_{G^tc}(x) do
          If y is not marked then
            begin
            mark y;
            s_{G^tc}(a_i):=s_{G^tc}(a_i)∪{y};
            end;
        end;
      S:=S\{x};
      end;
    For all y∈s_{G^tc}(a_i) do deletemarks (y);
    end;
End.
```

Definition 3.3. Let $G=(X, E)$ be an acyclic digraph. Let us denote by $M(G)$ the following expression:

$$\sum_{y\in X} \sum_{z\in \mathrm{Succ}_{G_{tr}}(y)} |\mathrm{Succ}_{G^{tc}}(z)|.$$

Remark. If we want to keep the notations used in the algorithm, we can express $M(G)$ in the following way

$$M(G)=\sum_{i\in[1,|X|]} \sum_{x\in \mathrm{Succ}_{G_{tr}}(a_i)} |\mathrm{Succ}_{G^{tc}}(x)|.$$

The next theorem shows that the complexity of the previous algorithm is a linear function of $M(G)$.

Theorem 3.4. *The top-down algorithm computes the transitive closure* $G^{tc}=(X, E^{tc})$ *and the transitive reduction* $G_{tr}=(X, E_{tr})$ *of a graph* $G=(X, E)$ *in* $O(n+M(G))$.

Proof. (a) Let us start by proving that the algorithm computes the transitive closure $G^{tc}=(X, E^{tc})$ and the transitive reduction $G_{tr}=(X, E_{tr})$ of G. We have to prove that at the end of the algorithm, for each x in X, $S_{G^{tc}}(x)=\mathrm{Succ}_{G^{tc}}(x)$ and $S_{G_{tr}}(x)=\mathrm{Succ}_{G_{tr}}(x)$. The proof of these two facts comes from the following conditions:

(i) $\forall x\in X$, $\mathrm{Succ}_{G_{tr}}(x)\subseteq \mathrm{Succ}_{G}(x)$;

(ii) '$S_{G_{tr}}(a_i)\subseteq S_{G^{tc}}(a_i)$' is invariant for the **While** loop;

(iii) 'min(L/S) is not marked if and only if it belongs to $\mathrm{Succ}_{G_{tr}}(a_i)$' is invariant for the **While** loop;

(iv) '$\forall j>i$, $S_{G^{tc}}(a_j)=\mathrm{Succ}_{G^{tc}}(a_j)$' is invariant for the second **For** loop;

The first one comes immediately from the definition. The second is easy, since in the **While** loop, if we add a vertex to $S_{G_{tr}}(a_i)$, we add it also to $S_{G^{tc}}(a_i)$. To be convinced of condition (iii), we have to remark that for $x \in \min(L/S(a_i))$, since L is a linear extension, there is no $a_j >_L x$ such that $x \in \mathrm{Succ}_{G^{tc}}(a_j)$, and if x is not marked, it is not a successor of an a_j with $a_i <_L a_j <_L x$ because, as the condition (iv) is assumed to be true for $j > i$, all the elements of $\mathrm{Succ}_{G^{tc}}(a_j)$ are marked; thus, by definition of S, $x \in \mathrm{Succ}_{G_{tr}}(a_i)$. For condition (iv), since x is greater than a_i in L, $S_{G^{tc}}(x) = \mathrm{Succ}_{G^{tc}}(x)$ and, so, with the previous conditions, it is clear that at the end of the first **For** loop, $S_{G^{tc}}(a_i) = \mathrm{Succ}_{G^{tc}}(a_i)$. Since the sets $S_{G^{tc}}(a_j)$ have not been changed for $j > i$, the condition is still checked. On the other hand, condition (iii) gives us that $S_{G_{tr}}(a_i) = \mathrm{Succ}_{G_{tr}}(a_i)$, which achieves the correctness of the algorithm.

(b) Let us now show that the algorithm runs in $O(n+M(G))$. Using Proposition 3.2, the cost of the preprocessing operation to obtain the L-ordered representation of G is in $O(n+m)$. So, we can assume that the 'Take $x \in \min(L/S)$' operation is in $O(1)$. It is clear that for a vertex $y \in S_{G^{tc}}(x)$ the operation 'mark' is in $O(1)$ and, so, for each x in $S_{G_{tr}}(a_i)$ the cost is in $O(|S_{G^{tc}}(x)|)$; thus, for each a_i, the cost of the **While** loop is in $O(\sum_{x \in S_{G_{tr}}(a_i)} |S_{G^{tc}}(x)|)$. Since the complexity of the last **For** loop is in $O(|\mathrm{Succ}_{G^{tc}}(a_i)|)$, the top-down algorithm works in $O(n+m+M(G))$. □

Unless the graph G contains many vertices having degrees 0, $O(n+m+M(G)) = O(M(G))$. Moreover, the behavior of top-down algorithm is completely characterized by an invariant $M(G)$ of the graph. As we are now going to see, for some classes of acyclic digraphs, some evaluations of $M(G)$ can be given.

3.2. Complexity results for the top-down algorithm

First of all, let us express $M(G)$ using the degrees in G_{tr} and G^{tc}. The following formula is an immediate reformulation of the original expression. In some cases it is useful to see $M(G)$ in this way.

Proposition 3.5. $M(G) = \sum_{x \in X} d^-_{tr}(x) \cdot d^+_{tc}(x) = \sum_{xy \in E_{tr}} d^+_{tc}(y)$.

From this expression, it can be immediately noticed that in the case of a chain, $M(G) \in O(n+m^{tc})$. In the case of a bipartite G, $M(G) = 0$. Furthermore, $M(G)$ is an invariant of the order associated with the input graph G.

Mehlhorn [10] used a model of random acyclic digraphs $G = (X, E)$, for which $ij \in E$, with $i < j$, is present with probability $\varepsilon(n)$. Moreover, the events $ij \in E$ are supposed to be independent. Then Mehlhorn proved that $d^-_{tr}(x) = O(\sqrt{n})$ and, thus, the top-down algorithm runs in $O(n^{5/2})$ for random digraphs. This bound was strengthened up to $O(n^2 \log n)$ by Simon [15]. Therefore, it is really useful to study and use this algorithm and also to evaluate $M(G)$ for particular classes of orders.

Proposition 3.6. *If G is an acyclic digraph such that the undirected graph associated with G_{tr} has no cycles,* i.e. *the order G^{tc} is the disjoint union of tree-like orders, then the top-down algorithm works in $O(n+m^{tc})$.*

Proof. Since it is clear that $\forall x \in X$, $\forall y, z \in \mathrm{Succ}_{G_{tr}}(x)$, $\mathrm{Succ}_{G^{tc}}(y) \cap \mathrm{Succ}_{G^{tc}}(z) = \emptyset$, we have

$$|\mathrm{Succ}_{G^{tc}}(x)| = |\mathrm{Succ}_{G_{tr}}(x)| + \sum_{z \in \mathrm{Succ}_{G_{tr}}(x)} |\mathrm{Succ}_{G^{tc}}(z)|$$

and the result follows. □

Proposition 3.7. *If G is an acyclic digraph such that the external or internal degree in G_{tr} of any vertex is bounded above by k, the top-down algorithm works in $O(n+m^{tc})$.*

Proof. (a) Assume that the external degree in G_{tr} is bounded by k: $\forall y \in X$, $|\mathrm{Succ}_{G_{tr}}(y)| \leqslant k$; so,

$$\sum_{z \in \mathrm{Succ}_{G_{tr}}(y)} |\mathrm{Succ}_{G^{tc}}(z)| \leqslant k \max_{z \in \mathrm{Succ}_{G_{tr}}(y)} |\mathrm{Succ}_{G^{tc}}(z)|.$$

Thus, $M(G) \leqslant k \cdot m^{tc}$ and the result follows.

(b) Concerning the internal degree, the result follows immediately from Proposition 3.5. □

Corollary 3.8. *If G is an acyclic digraph such that G^{tc} is an order of bounded width, the top-down algorithm works in $O(n+m^{tc})$.*

Proof. Immediate since the width of an order is at least equal to the maximum internal or external degree of a vertex in its transitive reduction. □

Remarks. We note that the same result cannot be obtained if G is an acyclic digraph with bounded height. Indeed, let H be the class of all the orders H_n obtained by the series composition of three antichains of size n (see Fig. 2). On this class $M(G) \in \Omega(n^3)$ and $m^{tc} \in O(n^2)$. So, the top-down algorithm is not linear in $n+m^{tc}$.

Furthermore, Fig. 3 shows a graph such that $M(G) \in O(n+m^{tc})$ and $M(G^-) \in O(n^3)$, where G^- is the dual graph of G obtained by reversing the direction of all arcs.

Theorem 3.9. *If P is a partial order with bounded decomposition width, the top-down algorithm works in $O(n+m^{tc})$ if the input is the canonical decomposition tree.*

Proof. Let $T(P)$ be the canonical decomposition tree of P. Let $N=(X_N, E_N)$ be an internal node such that $N = Q_{a_1}^{P_1} \cdots {}_{a_p}^{P_p}$, with $P_i=(X_i, E_i)$ and $Q=(X_Q, E_Q)$, and assume

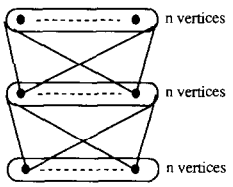

Fig. 2. A partially ordered set of the class H.

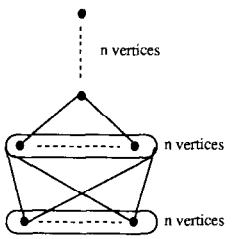

Fig. 3.

that for each P_i, we have its transitive closure $P_i^{tc}=(X_i, E_1^{tc})$. First of all, compute the transitive closure of Q with the top-down algorithm. For each $x\in X_i$,

$$\mathrm{Succ}_{N^{tc}}(x)=\left(\bigcup_{j/a_j\in \mathrm{Succ}_{Q^{tc}}(a_i)} X_j\right)\cup \mathrm{Succ}_{P_i^{tc}}(x).$$

Since, by definition of $T(P)$, the ground sets of the P_i are pairwise disjoints, these operations are disjoint unions, and, so, the complexity of building $\mathrm{Succ}_{N^{tc}}(x)\in O(|\mathrm{Succ}_{N^{tc}}(x)|-|\mathrm{Succ}_{P_i^{tc}}(x)|)$. Thus, the complexity of building the transitive closure of N is equal to the complexity of computing Q^{tc} plus $O(|X_Q|+\sum_{x\in X_N}(|\mathrm{Succ}_{N^{tc}}(x)|-|\mathrm{Succ}_{P_i^{tc}}(x)|))$. Obviously, if N is a series or a parallel node, the computation of Q^{tc} with the top-down algorithm is in $O(|X_Q|+|E_Q^{tc}|)$. On the other hand, if N is a prime node, as Q is of bounded width. using Corollary 3.8, the complexity is in $O(|X_Q|+|E_Q^{tc}|)$. So, the computation of N^{tc} can be done in $O(|X_N|+|E_N^{tc}|)$. The result follows immediately by induction. □

When the transitive closure of the order is given, one can use the $O(n^2)$ algorithm of Muller and Spinrad [12] to obtain the canonical decomposition tree. However, as far as we know, it is still an open problem to find this decomposition tree in $O(n^2)$ when the data is any graph between the transitive reduction and closure of the order.

Furthermore, it is possible to give an upper bound for the top-down algorithm as it is stated in Proposition 3.10. The proof of this proposition is immediate using the transformation of Fig. 4.

Proposition 3.10. *For any graph $G=(X, E)$ with rank decomposition $X_1, \ldots, X_r$ there exists a multipartite digraph $G'=(X_1, \ldots, X_r, E')$ such that $M(G)\leqslant M(G')$ and $E^{tc}\subseteq E'^{tc}$.*

In other words, the top-down algorithm has its worst cases on complete multipartite digraphs.

3.3. A more sophisticated use of top-down algorithm

In this section we are going to see that it is possible to use the top-down algorithm in a different way. The idea is to start from a class of acyclic digraphs in which the

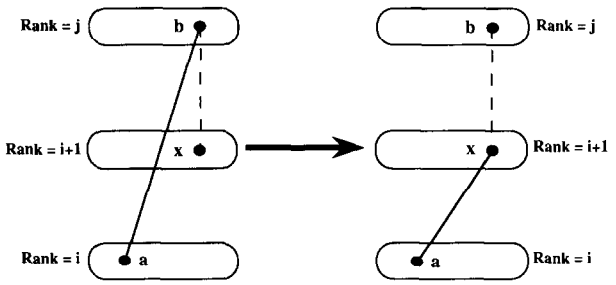

Fig. 4. For each edge $ab \in E_{tr}$ such that $a \in X_i$, $b \in X_j$, $j \neq i+1$, there exists $x \in (X_{i+1} \cap \mathrm{Pred}_{G^{tc}}(b)) \setminus \mathrm{Succ}_{G_{tr}}(a)$. The edge ab is deleted from E_{tr} and the edge ax is added to E_{tr}.

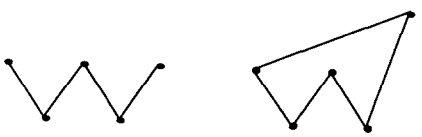

Fig. 5.

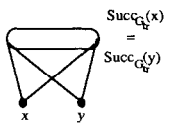

Fig. 6.

top-down algorithm is not linear and to transform linearly the input graph into a graph which has nearly the same transitive closure, and for which the top-down algorithm works in linear time. Let us consider a new class of acyclic digraphs:

Definition 3.11. An acyclic digraph $G=(X, E)$ is *strongly W-free* if and only if $\forall x, y \in X$, $\mathrm{Succ}_{G_{tr}}(x) \cap \mathrm{Succ}_{G_{tr}}(y) = \emptyset$ or $\mathrm{Succ}_{G_{tr}}(x) \subseteq \mathrm{Succ}_{G_{tr}}(y)$ or $\mathrm{Succ}_{G_{tr}}(y) \subseteq \mathrm{Succ}_{G_{tr}}(x)$, i.e. the direct successors sets cannot have a strict intersection.

It should be noted that the recognition of strongly W-free graphs can be done in linear time (for more details, see [11]).

It is easy to deduce from Definition 3.11 that an acyclic digraph $G=(X, E)$ is strongly W-free if and only if G_{tr} does not contain any subgraph isomorphic to one of the graphs of Fig. 5.

The following restriction will be done: the strongly W-free graph given as input is transitively reduced. So, we are just interested in the computation of its transitive closure.

This approach comes from an idea implicitly contained in [9], which is the following: if two vertices x and y of a graph G have the same direct successors set, we just have to compute once the union of the successors of the elements of $\mathrm{Succ}_{G_{tr}}(x) (= \mathrm{Succ}_{G_{tr}}(y))$ to have both $\mathrm{Succ}_{G^{tc}}(x)$ and $\mathrm{Succ}_{G^{tc}}(y)$ (see Fig. 6).

This idea can be generalized by noticing that if $\mathrm{Succ}_{G_{tr}}(x) \subseteq \mathrm{Succ}_{G_{tr}}(y)$ then

$$\mathrm{Succ}_{G^{tc}}(y) = \left(\bigcup_{z \in \mathrm{Succ}_{G_{tr}}(y) \setminus \mathrm{Succ}_{G_{tr}}(x)} \mathrm{Succ}_{G^{tc}}(z) \right) \cup \mathrm{Succ}_{G^{tc}}(x)$$

(see Fig. 7).

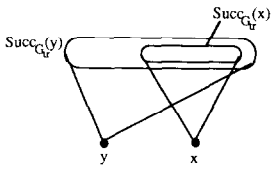

Fig. 7.

Fig. 8.

As will be shown in the sequel, this remark will be very useful in the case of strongly W-free graphs.

Let $G_{tr}=(X, E_{tr})$ be the transitive reduction of a strongly W-free graph. We are going to transform G_{tr} in a digraph $G'=(X, E')$. For this transformation, we need to consider an order $P=(X, <_P)=(X, E_P)$ associated with G_{tr} and satisfying the following conditions:

- $\mathrm{Succ}_{G_{tr}}(x)=\mathrm{Succ}_{G_{tr}}(y)$ implies that x and y are comparable;
- $\emptyset \neq \mathrm{Succ}_{G_{tr}}(y) \subset \mathrm{Succ}_{G_{tr}}(x) \Rightarrow x <_P y$

and P minimal for these conditions.

The existence of such an order is immediate (in fact, any order minimal for the conditions can be obtained from any other minimal one just by permuting the vertices with the same successors set). We can remark that if P is an order minimal for the conditions then P_{tr} does not contain any subgraph isomorphic to the graph of Fig. 8.

We can now define G' in the following way: $y \in \mathrm{Succ}_{G'}(x)$ if and only if one of the two following conditions is satisfied:

- $y \in \mathrm{Succ}_{G_{tr}}(x)$ and $\forall z \in X$, $(x \neq z$ and $\mathrm{Succ}_{G_{tr}}(z) \subset \mathrm{Succ}_{G_{tr}}(x)) \Rightarrow y \notin \mathrm{Succ}_{G_{tr}}(z)$;
- $y \in \mathrm{Succ}_{P_{tr}}(x)$.

This definition is illustrated on the example of Fig. 9.

We can immediately remark some conditions of the graph G' (where $x \|_G y$ means that x is incomparable with y in the order induced by G^{tc}).

Proposition 3.12. (i) *G' is transitively reduced.*

(ii) $\forall x, y \in X$, $x \|_{G'} y \Rightarrow x \|_G y$.

(iii) *G' does not contain any subgraph isomorphic to the graph of Fig. 8.*

(iv) $\forall x \in X$, $\mathrm{Succ}_{G^{tc}}(x)=(Succ_{G'}(x) \cup (\mathrm{Succ}_{G^{tc}}(Succ_{G'}(x)))) \setminus \mathrm{Succ}_{P_{tr}}(x)$.

Condition (iv) of this proposition shows that it is possible to obtain G^{tc} by applying to the graph G' a slightly modified version of the top-down algorithm (that will be called modified top-down) for which the operation

$$S_{G^{tc}}(a_i) := S_{G^{tc}}(a_i) \cup \{x\};$$

of the algorithm is not done if $x \in \mathrm{Succ}_{P_{tr}}(a_i)$. This modification, clearly, does not transform the complexity of the algorithm.

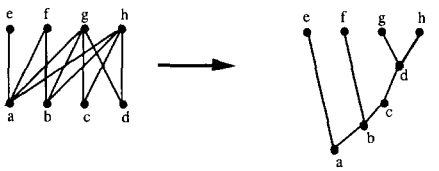

Fig. 9.

Using Proposition 3.6 and condition (iii) of Proposition 3.12, we can say that the top-down algorithm can be used to compute the transitive closure G'^{tc} of G' in $O(|X|+|E'^{tc}|)$ and, so, the modified top-down algorithm allows one to compute G^{tc} from G' in $O(|X|+|E^{tc}|)$.

So, we just have to show that the graph G' can be computed in $O(|X|+|E_{tr}|)$ from G_{tr}. To prove that, we are going to use an algorithm based on the following remarks:

- If the vertices of X are considered in the order $x_1, \ldots, x_n$ of growing external degree in G_{tr}, then this order respects the inclusion order of successors sets in G_{tr}. In other words, $i<j \Rightarrow (\mathrm{Succ}_{G_{tr}}(x_i) \cap \mathrm{Succ}_{G_{tr}}(x_j)=\emptyset$ or $\mathrm{Succ}_{G_{tr}}(x_i) \subseteq \mathrm{Succ}_{G_{tr}}(x_j))$.
- If X does not contain any vertex with the same successors sets in G_{tr} then, if $x_i \in X$, the vertex y such that $x_i \in \mathrm{Succ}_{P_{tr}}(y)$ (the unicity of this vertex comes from the previous condition) is the vertex verifying $y \in \mathrm{Pred}_{G_{tr}}(\mathrm{Succ}_{G_{tr}}(x_i)) \setminus \{x_1, \ldots, x_i\}$ and of external degree in G_{tr} minimal for this condition.

Therefore, the algorithm will work in the following way in order to compute two graphs $G_1=(X, E_1)$ and $G_2=(X, E_2)$ such that $G_2=P_{tr}$ and $G'=G_1 \cup G_2$:

- Initialize E_1 and E_2: $E_1:=E_{tr}$ and $E_2:=\emptyset$.
- Examine the vertices in the order $x_1, \ldots, x_n$ (if $y=x_i$ then i is the index of y) and for each vertex x_i such that there is no $j<i$ such that $\mathrm{Succ}_{G_{tr}}(x_i)=\mathrm{Succ}_{G_{tr}}(x_j)$ do:

• delete in E_1 all the edges zy where $y \in \mathrm{Succ}_{G_1}(x_i)$ and where $z(\neq x_i) \in \mathrm{Pred}_{G_1}(y)$;

• order linearly all the vertices z of $\mathrm{Pred}_{G_1}(\mathrm{Succ}_{G_1}(x_i))$ satisfying $d^+_{G_{tr}}(z)=d^+_{G_{tr}}(x_i)$, with x_i as the greatest element; add in E_2 all the covering edges of this total order; let u be the minimal element of the previous total order;

• search among the vertices of $\mathrm{Pred}_{G_1}(\mathrm{Succ}_{G_1}(x_i))$ the vertex t (if it exists) satisfying the following conditions:
  - $d^+_{G_{tr}}(t)>d^+_{G_{tr}}(x_i)$ and $d^+_{G_{tr}}(t)$ is minimum for these conditions;
  - the index of t is minimum for these conditions;

• if there is such a vertex t, and the edge tu to E_2.

It is clear that, after the execution of this algorithm, $G'=G_1 \cup G_2$. Let us now study it in more details in order to show its linearity.

Input: The transitive reduction of a strongly W-free acyclic digraph: $G_{tr}=(X, E_{tr})$. The vertices X are sorted by increasing external degree in G_{tr}: $x_1, \ldots, x_n$ (obviously, this ordering can be obtained in linear time); we also have a mapping *index* defined by: if $x=x_i$, then $index(x)=i$.

Output: The graph G' as defined previously.

Algorithm:

Begin
$i := 1$;
$E_1 := E_{tr}$;
$E_2 := \emptyset$;
{Let us note $G_1 = (x, E_1)$ and $G_2 = (x, E_2)$}
while $i \leqslant n$ **do**
begin
if x_i is not marked **and** $d^+_{G_{tr}}(x_i) \neq 0$ **then**
begin
{search among $\mathrm{Pred}_{G_1}(\mathrm{Succ}_{G_1}(x_i))$ a vertex t with $d^+_{G_{tr}}(t)$ minimum different from $d^+_{G_{tr}}(x_i)$ and of minimal index for this condition}
let $y \in \mathrm{Succ}_{G_1}(x_i)$;
min $:= n+1$;
ind $:= n+1$;
find := false;
for all $z \in \mathrm{Pred}_{G_1}(y)$ **do**
if $d^+_{G_{tr}}(z) \neq d^+_{G_{tr}}(x_i)$ **and** $d^+_{G_{tr}}(z) <$ min **and** $index(z) <$ ind
then
begin
find := true;
$t := z$;
min $:= d^+_{G_{tr}}(z)$;
ind $:= index(z)$;
end;
{if find = false, $\forall z \in \mathrm{Pred}_{G_1}(\mathrm{Succ}_{G_1}(x_i))$, $\mathrm{Succ}_{G_1}(z) = Succ_{G_1}(x_i)$}
{now delete from E_1 the edges of G_{tr} which are not edges of G'}
for all $y \in \mathrm{Succ}_{G_1}(x_i)$ **do**
for all $z \neq x_i \in \mathrm{Pred}_{G_1}(y)$ **do** $E_1 := E_1 \setminus \{zy\}$;
{add the edges of P to E_2}
let $y \in \mathrm{Succ}_{G_1}(x_i)$;
for all $z \neq x_i \in \mathrm{Pred}_{G_1}(y)$ **do**
begin
$u := x_i$;
if $d^+_{G_{tr}}(z) = d^+_{G_{tr}}(x_i)$ **then**
begin
$E_2 := E_2 \cup \{zu\}$;
$u := z$;
mark z;
end;
end;
if find **then** $E_2 := E_2 \cup \{tu\}$;
end;
$i := i+1$;
end;
$G' := G_1 \cup G_2$;
End.

To be convinced of the linearity of the algorithm, we can first remark that each edge examined during the computation of a vertex x_i (this computation will be called step i) will never be examined later. In other words, we can associate with each vertex x_i the set of all edges examined during step i, and two such sets are necessarily disjoint. Indeed, let us characterize edges examined during step i. These are only edges from a vertex of $\mathrm{Pred}_{G_1}(\mathrm{Succ}_{G_1}(x_i))$ to a vertex of $\mathrm{Succ}_{G_1}(x_i)$; but except the edges from x_i to an element of $\mathrm{Succ}_{G_1}(x_i)$, all these edges are deleted from G_1 during step i and, thus, they cannot be examined once more. For the edges from x_i to an element of $\mathrm{Succ}_{G_1}(x_i)$ they could not be examined once more; indeed, the unique predecessor in G_1 (changed by the edges removed) of an element of $\mathrm{Succ}_{G_1}(x_i)$ is the vertex x_i and this vertex will never be checked again. On the one hand, as during step i, an edge is examined three times, it is clear that at the end of the algorithm, the number of operations needed for the examination of the edges of E_{tr} is linear in $|E_{\mathrm{tr}}|$. On the other hand, as it is clear that during each step the number of edges added to E_2 is less than the number of edges examined, the algorithm is linear in $|X|+|E_{\mathrm{tr}}|$.

Now we can state the following theorem:

Theorem 3.13. *Let $G=(X, E)$ be a strongly W-free acyclic digraph: it is possible to compute $G^{\mathrm{tc}}=(X, E^{\mathrm{tc}})$ from $G_{\mathrm{tr}}=(X, E_{\mathrm{tr}})$ in $O(|X|+E^{\mathrm{tc}}|)$.*

Remark. As an N-free order is strongly W-free, the previous theorem generalizes the result of Ma and Spinrad [9] for N-free orders [9].

A very natural question comes from this study: Is it possible to find a simple characterization of the class of acyclic digraphs for which the complexity of the top-down algorithm is linear? The examples of Figs. 2 and 3, which seem to represent 'worst cases', could suggest that the absence of complete bipartite graphs of sufficiently large size, as subgraphs in the transitive reduction of the graphs of the considered class, is a sufficient condition. The following proposition proved by Bollobàs in [3] as a corollary of a Reiman's theorem invalidates this idea

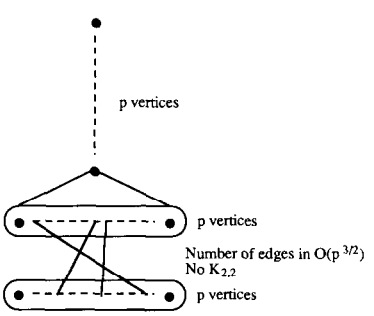

Fig. 10.

Proposition 3.14. (Bollobas [3]). *If p is sufficiently large, the maximum number of edges of a bipartite graph, composed of two levels of p vertices, without $K_{2,2}$ (denoted as $z(p; 2)$) satisfies that $p^{3/2} - p^{4/3} < z(p; 2) \leqslant \frac{1}{2}p + p\sqrt{4p-3}$; further $\lim_{p \to \infty} z(p; 2)p^{-3/2} = 1$.*

So, for each p sufficiently large, there exists a bipartite graph built up with two levels of p vertices, without $K_{2,2}$ and for which the number of edges is in $O(p^{3/2})$. Thus, let us consider such a bipartite graph and let us 'put it' using a series operation 'under' a chain of p vertices (see Fig. 10). For the graph obtained, we have $M(G) \in O(n^{5/2})$ with $n = 3p$ and, so, the algorithm is not linear on this class.

References

[1] A.V. Aho, M.R. Garey and J.D. Ullman, The transitive reduction of a directed graph, SIAM J. Comput. 1, 2 (1972) 131–137.

[2] V.L. Arlazarov, E.A. Dinic, M.A. Kronrod and I.A. Faradzev, On economical construction of the transitive closure of an oriented graph, Dokl. Akad. Nauk SSSR 194(3), Sov. Math. Dokl. 11 (1970) 1209–1210.

[3] B. Bollobàs, Extremal Graph Theory (Academic Press, New York, 1978).

[4] J.P. Bordat, Complexité de problèmes liés aux graphes sans circuits, Informatique Théorique et Applications 21 (1987) 181–197.

[5] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, Proc. 19th Ann. Symp. on the Theory of Computation (1987) 1–6.

[6] M.J. Fisher and A.R. Meyer, Boolean matrix multiplication and transitive closure, Conference Record, Proc. 12th Ann. Symp. on Switching and Automata Theory (1971) 129–131.

[7] M.E. Furman, Application of a method of fast multiplication of matrices in the problem of finding the transitive closure of a graph, Dokl. Akad. Nauk. SSSR 194(3), Sov. Math. Dokl. 11 (1970) 1252.

[8] A. Goralcikova and V. Koubek, A reduct and closure algorithm for graphs, Math. Foundations Comput. Sci. (1979) 301–307.

[9] T.H. Ma and J. Spinrad, Transitive closure for restricted classes of partial orders, in: Proc. WG 90, Lectures Notes in Computer Science (Springer, Berlin, 1991).

[10] K. Mehlhorn, Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness (Springer, Berlin, 1984).

[11] M. Morvan, Algorithmes linéaires et invariants d'ordres, Thèse de doctorat, Universitié de Montpellier II, 1991.

[12] J.H. Muller and J. Spinrad, Incremental modular decomposition, J. ACM 36 (1989) 1–19.

[13] P. Purdom Jr, A transitive closure algorithm, Bit 10 (1970) 76–94.

[14] C.P. Schnorr, An algorithm for transitive closure with linear expected time, SIAM J. Comput. 7 (1978) 127–133.

[15] K. Simon, An improved algorithm for transitive closure on acyclic digraphs, Theoret. Comput. Sci. 58 (1988) 325–346.

[16] K. Simon, Finding a minimal transitive reduction in a strongly connected digraph within linear time, Proc. WG 89, Lectures Notes in Computer Science, Vol. 411 (Springer, Berlin, 1990) 245–259.

[17] J. Valdes, R.E. Tarjan and E.L. Lawler, The recognition of series–parallel digraphs, SIAM J. Comput. 11 (1982) 298–313.

[18] S. Warshall, A theorem on boolean matrices (Computer Associates, Massachussets (1962) 11–12.