Theoretical Computer Science 58 (1988) 325–346
North-Holland

AN IMPROVED ALGORITHM FOR TRANSITIVE CLOSURE ON ACYCLIC DIGRAPHS

Klaus SIMON

Fachbereich 10, Angewandte Mathematik und Informatik, Universität des Saarlandes, 6600 Saarbrücken, Fed. Rep. Germany

Abstract. In [6] Goralčíková and Koubek describe an algorithm for finding the transitive closure of an acyclic digraph G with worst-case runtime $O(n \cdot e_{red})$, where n is the number of nodes and e_{red} is the number of edges in the transitive reduction of G. We present an improvement on their algorithm which runs in worst-case time $O(k \cdot e_{red})$ and space $O(n \cdot k)$, where k is the width of a chain decomposition. For the expected values in the $G_{n,p}$ model of a random acyclic digraph with $0 < p < 1$ we have

$$E(k) = O\left(\frac{\log(p \cdot n)}{p}\right), \qquad E(e_{red}) = O(n \cdot \log n),$$

$$E(k \cdot e_{red}) = \begin{cases} O(n^2) & \text{for } \log^2 n/n \leq p < 1, \\ O(n^2 \cdot \log\log n) & \text{otherwise,} \end{cases}$$

where "log" means the natural logarithm.

1. Introduction

A directed graph $G = (V, E)$ consists of a vertex set $V = \{1, 2, 3, \ldots, n\}$ and an edge set $E \subseteq V \times V$. Each element (v, w) of E is an edge and joins v to w. If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are directed graphs, G_1 is a subgraph of G_2 if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. The subgraph of G_2 induced by the subset V_1 of V_2 is the graph $G_1 = (V_1, E_1)$, where E_1 is the set of all elements of E_2 which join pairs of elements of V_1. Unless we specify otherwise, any subgraph referred to in this paper is the subgraph induced by its vertex set. A path in a graph from vertex v_0 to vertex v_s is a sequence of vertices $v_0, v_1, \ldots, v_s$ such that (v_{i-1}, v_i) is an edge for $i \in \{1, 2, \ldots, s\}$; s is the length of the path. The path is simple if all its vertices are pairwise distinct. A path $v_0, \ldots, v_s$ is a cycle if $s \geq 1$ and $v_0 = v_s$ and a simple cycle if in addition $v_1, \ldots, v_{s-1}$ are pairwise distinct. A graph without cycles is acyclic. A topological sorting of a digraph $G = (V, E)$ is a mapping ord: $V \to \{1, 2, \ldots, |V|\}$ such that for all edges $(v, w) \in E$ we have $\text{ord}(v) < \text{ord}(w)$. The relation between an acyclic digraph G and a topological sorting is given in Theorem 1.1 (for a proof see [8, pp. 320–323]).

Theorem 1.1. *$G = (V, E)$ is acyclic if and only if it has a topological sorting. A topological sorting of an acyclic graph can be computed in linear time* $O(|V| + |E|)$.

Throughout this paper we will assume that the acyclic digraph $G = (V, E)$ is topologically sorted, i.e., $(i, j) \in E$ implies $i < j$ and that the adjacency lists $\text{out}(v) =$

0304-3975/88/$3.50

$\{w \in V \mid (v, w) \in E\}$ are sorted in increasing order. This can be achieved in linear time $O(|V|+|E|)$ (see [8, p. 323]). Now we need some more definitions. The node w is in the reflexive, transitive closure $\text{out}^*(v)$ if and only if there is a path from v to w in G. The set $\text{out}^{\text{red}}(v) = \{w \in \text{out}(v) \mid \text{there is no path of length at least 2 from } v \text{ to } w \text{ in } G\}$ is called the transitive reduction of the node v. Let $E^* = \{(v, w) \mid w \in \text{out}^*(v)\}$, $E_{\text{red}} = \{(v, w) \mid w \in \text{out}^{\text{red}}(v)\}$; then $G^* = (V, E^*)$ ($G_{\text{red}} = (V, E_{\text{red}})$) is called the transitive closure (transitive reduction) of G. Further we use the usual notation $e = |E|$, $e^* = |E^*|$, $e_{\text{red}} = |E_{\text{red}}|$, $n = |V|$, $\gamma(v) = |\text{out}(v)|$, $\gamma^*(v) = |\text{out}^*(v)|$, and $\gamma^{\text{red}}(v) = |\text{out}^{\text{red}}(v)|$.

Let $\text{out}(v) = \{w_1 < \cdots < w_s\}$; then it is easy to see that we obtain

$$\text{out}^*(v) = \{v\} \cup \text{out}^*(w_1) \cup \text{out}^*(w_2) \cup \cdots \cup \text{out}^*(w_s). \tag{1}$$

For a topologically sorted digraph $G = (V, E)$ the observation that

$$\begin{aligned} w_i \notin \text{out}^{\text{red}}(v) \;&\Leftrightarrow\; w_i \in \text{out}^*(w_1) \cup \cdots \cup \text{out}^*(w_{i-1}) \\ &\Leftrightarrow\; \text{out}^*(w_i) \subseteq \text{out}^*(w_1) \cup \cdots \cup \text{out}^*(w_{i-1}) \end{aligned} \tag{2}$$

was made by Goralčíková and Koubek to show $G^* = G^*_{\text{red}}$. Through negation from (2) we get

$$\text{out}^{\text{red}}(v) = \{w_i \in \text{out}(v) \mid w_i \notin \text{out}^*(w_1) \cup \cdots \cup \text{out}^*(w_{i-1})\} \tag{3}$$

and further

$$\bigcup_{w \in \text{out}(v)} \text{out}^*(w) = \bigcup_{z \in \text{out}^{\text{red}}(v)} \text{out}^*(z). \tag{4}$$

Here we give their algorithm.

Algorithm A (*Goralčíková, Koubek* [6])
Input: $G = (V, E)$
Output: $\text{out}^*(v)$ and $\text{out}^{\text{red}}(v)$, $\forall v \in V$

```
for v ← n downto 1 (* V = {1, ..., n} *)
do
   out*(v) ← {v};
   out^red(v) ← ∅;
   for ∀w ∈ out(v) (* in increasing order *)
   do
      if w ∉ out*(v)
      then
         out*(v) ← out*(v) ∪ out*(w);
         out^red(v) ← out^red(v) ∪ {w};
      fi
   od
od
```

Inside loop (2)–(13) we use a bitvector for $out^*(v)$. So the test "$w \in out^*(v)$" takes time O(1). Outside loop (2)–(13) the set $out^*(v)$ is kept as a linear list. This implies that the operation $out^*(v) \cup out^*(w)$ has execution time $O(|out^*(w)|) = O(n)$ and so we need total time $O(n \cdot e_{red})$. We will show later that the expected execution time is $O(n^2 \cdot \log n)$ for this algorithm. In this paper we give a better method for computing $out^*(v) \cup out^*(w)$. Our improvement is based on an efficient data structure, the so-called chain decomposition.

2. The algorithm

Definition 2.1. Let $G = (V, E)$ be an acyclic digraph. A partition $Z_1, \ldots, Z_k$ of V ($Z_i \neq \emptyset$ for $1 \leq i \leq k$ and $Z_1 \cup \ldots \cup Z_k = V$) is called a *chain decomposition* of $G = (V, E)$ if and only if every Z_i, $1 \leq i \leq k$, is a path in G. Because G is topologically sorted, we obtain for a path $Z_i = \{v_1 < \cdots < v_s\}$

$$(v_j, v_{j+1}) \in E \quad \forall 1 \leq j \leq k.$$

The integer k is called the *width* of the decomposition. A chain decomposition $Z_1, \ldots, Z_k$ induces the maps id, niv, and niv_j. Let $v \in V$, $A \subseteq V$, $i, j \in \{1, \ldots, k\}$; then

$$\text{id}(v) = i \iff v \in Z_i,$$

$$\text{niv}_j(A) = \min(A \cap Z_j), \qquad \text{niv}(A) = \{\text{niv}_j(A) \mid 1 \leq j \leq k\}.$$

In particular, we use the notation $\text{niv}_j(v)$ for $\text{niv}_j(out^*(v))$ and $\text{niv}(v)$ for $\text{niv}(out^*(v))$. In this paper we have only $A = out^*(w_1) \cup \cdots \cup out^*(w_s)$ and therefore $A \cap Z_j$ is a path in G with first node $\text{niv}_j(A)$ (see Fig. 1).

Now we come to the question how we can use a chain decomposition to speed up the computation of the transitive closure? The critical observation is given in Theorem 2.2.

Theorem 2.2. $\forall v \in V$:

$$out^*(v) = \bigcup_{1 \leq j \leq k} \{w \in Z_j \mid w \geq \text{niv}_j(v)\}.$$

Proof. Let $v \in V$ and let $Z_1, \ldots, Z_k$ be a chain decomposition. Then $Z_1, \ldots, Z_k$ are a partition of V and we have $Z_1 \cup \cdots \cup Z_k = V$. Since $out^*(v) \subseteq V$, we get

$$out^*(v) = \bigcup_{1 \leq j \leq k} (out^*(v) \cap Z_j).$$

Let Z_j be a path $v_1, \ldots, v_s$ with $v_1 < v_2 < \cdots < v_s$, $\text{niv}_j(v) = \min(out^*(v) \cap Z_j)$ and $out^*(v) \cap Z_j \neq \emptyset$. Now $\text{niv}_j(v)$ is an element of $out^*(v)$ and therefore the set

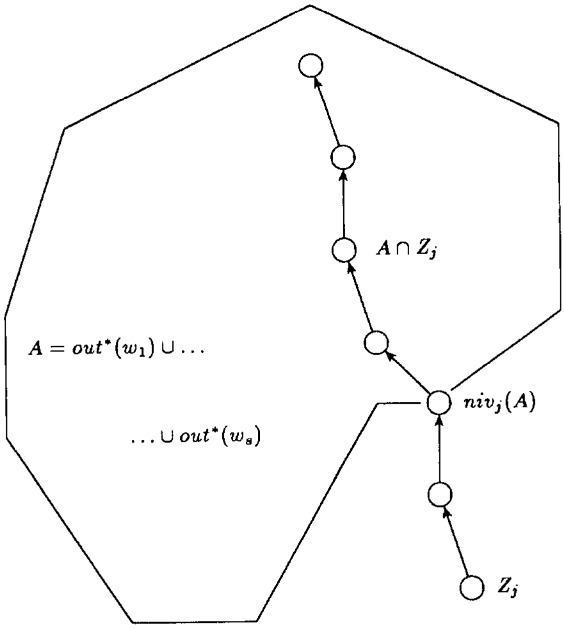


Fig. 1.

$(\text{out}^*(v) \cap Z_j)$ is a tail of Z_j, namely $v_l, v_{l+1}, \ldots, v_s$, where $v_l = \text{niv}_j(v)$. Then we have

$$\text{out}^*(v) = \bigcup_{1 \leq j \leq k} (\text{out}^*(v) \cap Z_j) = \bigcup_{1 \leq j \leq k} \{w \in Z_j \mid w \geq \text{niv}_j(v)\}. \qquad \square$$

This theorem shows that it is sufficient to compute $\text{niv}_j(v)$ for all $1 \leq j \leq k$ since the computation of the set $\{w \in Z_j \mid w \geq \text{niv}_j(v)\}$ is trivial for a given chain Z_j. In the following we will use the convention $\min(\emptyset) = \infty$. Now we describe relations that we need for the efficient computation of $\text{niv}(v)$. Let

$$\text{niv}_j(w_1) = \text{niv}_j(\text{out}^*(w_1)), \ldots, \text{niv}_j(w_s) = \text{niv}_j(\text{out}^*(w_s))$$

are given; then we infer

$$\begin{aligned}
&\text{niv}_j(\text{out}^*(w_1) \cup \cdots \cup \text{out}^*(w_s)) \\
&\quad = \min((\text{out}^*(w_1) \cup \cdots \cup \text{out}^*(w_s)) \cap Z_j) \\
&\quad = \min((\text{out}^*(w_1) \cap Z_j) \cup \cdots \cup (\text{out}^*(w_s) \cap Z_j)) \\
&\quad = \min(\min(\text{out}^*(w_1) \cap Z_j), \ldots, \min(\text{out}^*(w_s) \cap Z_j)) \\
&\quad = \min(\text{niv}_j(\text{out}^*(w_1)), \ldots, \text{niv}_j(\text{out}^*(w_s))) \\
&\quad = \min(\text{niv}_j(w_1), \ldots, \text{niv}_j(w_s)). \qquad (5)
\end{aligned}$$

Now we combine expression (5) with (1) and this leads us to Theorem 2.3.

Theorem 2.3. $\forall v \in V, \forall j, 1 \leq j \leq k$:

$$\text{niv}_j(v) = \begin{cases} \min(\{\text{niv}_j(w) \mid w \in \text{out}(v)\}) & \text{if } j \neq \text{id}(v), \\ v & \text{otherwise.} \end{cases}$$

Proof. Let $\text{out}(v) = \{w_1, \ldots, w_s\}$; then we get

$$\begin{aligned} \text{niv}_j(v) &= \text{niv}_j(\text{out}^*(v)) \\ &\overset{(1)}{=} \text{niv}_j(\{v\} \cup \underbrace{\text{out}^*(w_1) \cup \cdots \cup \text{out}^*(w_s)}_{=_{\text{def}} A}) \\ &= \min((\{v\} \cup A) \cap Z_j) \\ &= \min((\{v\} \cap Z_j) \cup (A \cap Z_j)) \\ &= \min(\min(\{v\} \cap Z_j). \underbrace{\min(A \cap Z_j))}_{=\text{niv}_j(A)} \\ &\overset{(5)}{=} \min(\min(\{v\} \cap Z_j), \min(\text{niv}_j(w_1), \ldots, \text{niv}_j(w_s))) \\ &= \begin{cases} \min(\text{niv}_j(w_1), \ldots, \text{niv}_j(w_s)) & \text{if } v \notin Z_j, \\ v & \text{otherwise.} \end{cases} \quad \square \end{aligned}$$

With Theorem 2.3 it is clear how we compute $\text{niv}(v) = \text{niv}(\text{out}^*(v))$. In Algorithm A we replace the operation

$$\text{out}^*(v) \leftarrow \text{out}^*(v) \cup \text{out}^*(w)$$

by

$$\text{niv}(\text{out}^*(v)) \leftarrow \text{niv}(\text{out}^*(v) \cup \text{out}^*(w)).$$

With expression (5) this reduces the execution time from

$$O(|\text{out}^*(w)|) \overset{(\text{L.3.5})}{=} O(n)$$

to

$$O(|\text{niv}(\text{out}^*(w))|) = O(k).$$

In general we find that k is very much smaller as $|\text{out}^*(w)|$ (see Section 3). Now it remains the problem how we implement the test "$w \in \text{out}^*(v)$" from Algorithm A. If we want to use our new data structure, we cannot realize set $\text{out}^*(v)$ as a bitvector. But now we use an array of integers for set $\text{niv}(\text{out}^*(v))$ and Lemma 2.4 shows this is sufficient.

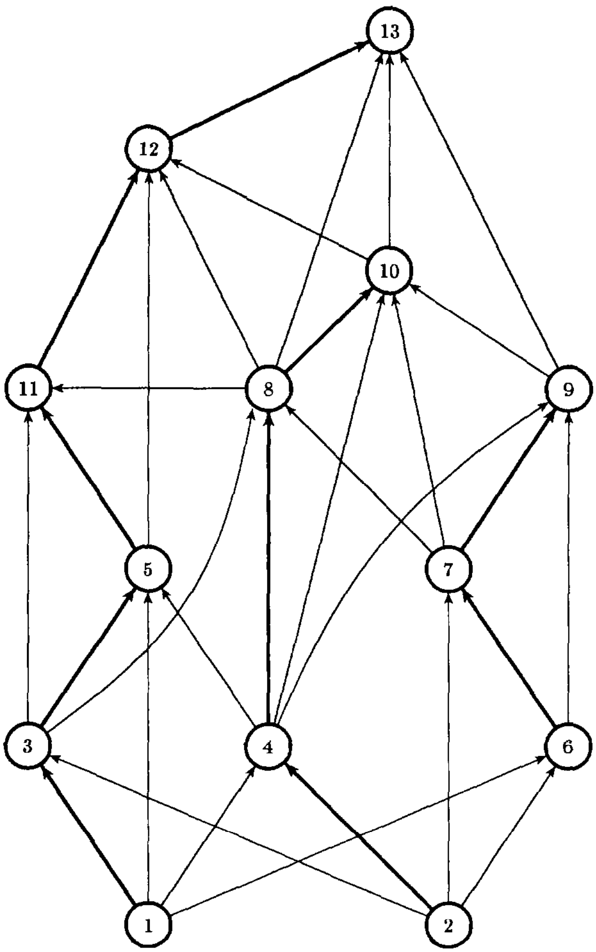
Lemma 2.4. *Let $v, w_1, \ldots, w_s \in V$ and $w \in Z_j$ ($\Leftrightarrow \text{id}(w) = j$). Then there is*

$$w \notin \text{out}^*(w_1) \cup \cdots \cup \text{out}^*(w_s) \Leftrightarrow w < \min(\text{niv}_{\text{id}(w)}(w_1), \ldots, \text{niv}_{\text{id}(w)}(w_s)).$$

Proof. With $A = \text{out}^*(w_1) \cup \cdots \cup \text{out}^*(w_s)$ we get

$$w \in A \Leftrightarrow w \in A \cap Z_{\text{id}(w)}$$
$$\Leftrightarrow w \geqslant \min(A \cap Z_{\text{id}(w)})$$
$$\Leftrightarrow w \geqslant \text{niv}_{\text{id}(w)}(A)$$
$$\overset{(5)}{\Leftrightarrow} w \geqslant \min(\text{niv}_{\text{id}(w)}(w_1), \ldots, \text{niv}_{\text{id}(w)}(w_s)).$$

Our claim is inferred through negation. □



$Z_1 = \{1, 3, 5, 11, 12, 13\}$

$Z_2 = \{2, 4, 8, 10\}$

$Z_3 = \{6, 7, 9\}$

Fig. 2. An example for chain decomposition.

A chain decomposition of an acyclic graph is easily constructed in time $O(n+e)$. Now we give one particular algorithm. In a greedy manner we find a first path Z. We remove Z from $G=(V, E)$ and restart the method (see Fig. 2).

Now have a look at Algorithm B. At line (4) V_i is realized as a bitvector S. Line (6) is implemented by sliding a pointer pt across bitvector S. All elements to the left of pt are not in V_i. The sets Z_i are kept as linear lists. Then loop (7)-(13) takes time $O(\gamma(v))$ for fixed v. Hence, the total time (1)-(18) is $O(n+e)$. Note that the total time spent in line (6) is $O(n)$ since pt is slid one across vector S.

Algorithm B (*computation of a chain decomposition*)
Input: $G=(V, E)$
Output: $Z_1, \ldots, Z_k$, id

```
(1)   i ← 1
(2)   for ∀v ∈ V do id[v] ← 0 od
(3)   V_i ← V
(4)   while V_i ≠ ∅
(5)   do
(6)      x ← min(V_i)
(7)      Z ← {x}
(8)      while ∃y ∈ V_i with (x, y) ∈ E
(9)      do
(10)        let y be minimal with y ∈ V_i and (x, y) ∈ E
(11)        Z ← Z + {y}
(12)        x ← y
(13)  od
(14)     Z_i ← Z
(15)     V_{i+1} ← V_i − Z_i
(16)     for ∀v ∈ Z_i do id[v] ← i od
(17)     i ← 1 + 1
(18)  od
```

In the following when we speak about a chain decomposition we will mean the decomposition constructed by this algorithm and we define the width $k=k(G)$ of the chain decomposition by

$$k \overset{\text{def}}{=} \max(\{s \in \mathbb{N} \mid Z_s \neq \emptyset\}).$$

Now we can compute $\text{niv}(v)$ for all $v \in V$. In Algorithm C we use the linear list nivlist$[v]$ for keeping $\text{niv}(v)$; nivfield, id are arrays of integers. So we come to Algorithm C.

Algorithm C (*computation of* $\text{niv}(v)$)
Input: $G=(V, E)$, $\text{id}: V \to \{1, \ldots, k\}$

Output: niv(v), out^{red}(v), $\forall v \in V$

```
 (1)  for s ← 1 to k do nivfield [s] ← ∞ od
 (2)  for v ← n downto 1
 (3)  do
 (4)     out^red[v] ← ∅; nivlist [v] ← ∅;
 (5)     for ∀w ∈ out(v) (* in increasing order *)
 (6)     do
 (7)     if w < nivfield[id[w]]
 (8)        then
 (9)           out^red[v] ← out^red[v] ∪ {w}
(10)           for ∀p ∈ nivlist[w]
(11)           do
(12)              nivfield[id[p]] ← min(nivfield[id[p]], p)
(13)           od
(14)        fi
(15)     od
(16)     nivfield[id[v]] ← v;
(17)     for s ← 1 to k
(18)     do
(19)        if nivfield[s] ≠ ∞
(20)        then nivlist[v] ← nivlist[v] ∪ nivfield[s] fi
(21)        nivfield[s] ← ∞
(22)     od
(23)  od
```

Correctness is shown by induction, starting with $v = n$. In particular we get, after every execution of loop (2)-(23),

$$\text{nivlist}[v] = \text{niv}(v) \quad \text{and} \quad \text{out}^{\text{red}}[v] = \text{out}^{\text{red}}(v).$$

($v = n$): Since loop (5)-(15) is not executed, we obtain $\text{nivlist}[n] = \{n\} = \text{niv}(n)$.
($v < n$). Now the induction hypothesis is

$$\forall w,\ v < w \leqq n: \quad \text{nivlist}[w] = \text{niv}(w).$$

There is, after line (13),

$$\forall p,\ p \in \text{nivlist}[w]: \quad \text{nivfield}[\text{id}[p]]_{\text{new}} = \min(\text{nivfield}[\text{id}[p]]_{\text{old}}, p).$$

With the induction hypothesis this is equivalent to

$$\forall j,\ 1 \leq j \leq k;\ \text{nivfield}[j]_{\text{new}} = \min(\text{nivfield}[j]_{\text{old}}, \text{niv}_j(w)) \qquad (*)$$

since

$$p = \text{niv}_{\text{id}(p)}(w) \quad \forall p \in \text{nivlist}[w] \overset{\text{(I.H.)}}{=} \text{niv}(w).$$

We infer inductively from (*) that before the execution of loop (6)–(15) it is valid that

$$\text{nivfield}[j] = \min(\{\text{niv}_j(z) \mid z \in \text{out}^{\text{red}}(v) \wedge z < w\}). \qquad (**)$$

Consequently, line (7) is executed if and only if

$$w < \text{nivfield}[\text{id}[w]] \overset{(**)}{\Leftrightarrow} w < \text{niv}_{\text{id}(w)}(v) \overset{(\text{L.2.4})}{\Leftrightarrow} w \in \text{out}^{\text{red}}(v).$$

This shows the correct construction of out^{red}. With induction on the number of executions of loop (6)–(15) we get from (**) after line (16) $\forall j$, $1 \leq j \leq k$:

$$\text{nivfield}[j] = \begin{cases} \min(\{\text{niv}_j(w) \mid w \in \text{out}^{\text{red}}(v)\}) & \text{if } j \neq \text{id}(v), \\ v & \text{otherwise} \end{cases}$$

and this leads with Theorem 2.3 to

$$\text{nivfield}[j] = \text{niv}_j(v) \quad \forall j,\ 1 \leq j \leq k.$$

This ends the correctness proof of Algorithm C.

Running time: Outside lines (6)–(15) the cost of the algorithm is clearly $O(e + n \cdot k)$. One execution of the loop (10)–(13) has cost $O(k)$ and this loop is executed only for $(v, w) \in E_{\text{red}}$. Hence, for Algorithm C we have total cost

$$O(e + n \cdot k) + O(e_{\text{red}} \cdot k) = O(e + e_{\text{red}} \cdot k).$$

Theorem 2.5. *The improved algorithm computes the transitive closure of an acyclic digraph in time* $O(e^* + e_{\text{red}} \cdot k)$.

Proof. Running time of the decomposition algorithm is $O(n + e)$. The computation of $\text{niv}(v)$, $\forall v \in V$ needs time $O(e + e_{\text{red}} \cdot k)$. From a chain decomposition of $Z_1, \ldots, Z_k$ with given $\text{id}(v)$, $\text{niv}(v)$, $\forall v \in V$, it is now trivial to compute E^* in time $O(e^*)$ (recall Theorem 2.2). Hence, we get a total running time

$$O(n + e) + O(e + e_{\text{red}} \cdot k) + O(e^*) = O(e^* + e_{\text{red}} \cdot k). \qquad \square$$

3. Average case

For the average case analysis we use the $G_{n,p}$ model of a random acyclic digraph with vertex set $\{1, \ldots, n\}$ in which the possible edges (i, j), $1 \leq i < j \leq n$, occur independently with probability p, $0 < p < 1$. An introduction to the theory of random graphs was given by Erdös and Spencer in [3]. By this model the size of k, e_{red}, or $k \cdot e_{\text{red}}$ is a random variable. Our aspiration is to obtain good upper bounds for the expected values of these random variables, especially for the product $k \cdot e_{\text{red}}$. Note that the latter is a product of two dependent random variables and its analysis takes a lot of time. Therefore we give the main findings first and see the proofs and other results later in their logical order. We write $\Pr(A)$ for the probability of event A

and further $\Pr(A|B)$ for the probability of A on condition of event B. Let X be a random variable; then $E(X)$ means the expected value of X. "log" stands for the natural logarithm.

$$\text{(Lemma 3.4)} \qquad E(k) \leqslant \frac{\log(p \cdot n)}{p} + 1,$$

$$\text{(Corollary 3.8)} \qquad E(e_{\text{red}}) \leqslant n \cdot (\log n + 2),$$

$$\text{(Lemma 3.12)} \qquad E(k \cdot e_{\text{red}}) \leqslant 4 \cdot \frac{\log n}{p} \cdot E(e_{\text{red}}) + 1,$$

$$\text{(Lemma 3.13)} \qquad E(k \cdot e_{\text{red}}) \leqslant (E(k) + 1) \cdot E(e).$$

Theorem 3.1. *By use of the $G_{n,p}$ model of a random digraph our Algorithm* C *computes the transitive closure of an acyclic digraph in expected time*:

$$O(n^2) \quad \textit{for} \frac{\log^2 n}{n} \leqslant p < 1 \quad \textit{and}$$

$$O(n^2 \cdot \log\log n) \quad \textit{for } 0 < p < \frac{\log^2 n}{n}.$$

Proof. For $p \geqslant (\log^2 n)/n$ we use Lemma 3.12 and otherwise Lemma 3.13.

Case $p \geqslant (\log^2 n)/p$:

$$\begin{aligned} E(e_{\text{red}}) &\leqslant 4 \cdot \frac{\log n}{p} \cdot E(e_{\text{red}}) + 1 \\ &\leqslant 4 \cdot \frac{\log n}{(\log^2 n)/n} \cdot n \cdot (\log n + 2) + 1 \\ &\leqslant O\left(n^2 \cdot \frac{\log^2 n}{\log^2 n}\right) \leqslant O(n^2). \end{aligned}$$

Case $p \leqslant (\log^2 n)/n$:

$$\begin{aligned} E(k \cdot e_{\text{red}}) &\leqslant (E(k)+1) \cdot E(e) \leqslant O\left(\frac{\log(p \cdot n)}{p} \cdot p \cdot n^2\right) \\ &\leqslant O\left(n^2 \cdot \left(\log\left(\frac{\log^2 n}{n} \cdot n\right)\right)\right) \leqslant O(n^2 \cdot (2 \cdot \log(\log n))) \\ &\leqslant O(n^2 \cdot \log\log n). \qquad \square \end{aligned}$$

In the random graph $G_{n,p}$ we describe the size of the chain decomposition $k = k(G_{n,p}) = k_n$ and the size of the transitive closure of the first vertex $\gamma^*(1) = \gamma^*(1)(G_{n,p}) = \gamma_n^*$ as a Markov chain with discrete time $t = n$ [4]. First we consider the behaviour of k at the point of transition from n to $n+1$. If $k_n = l$, $1 \leqslant l \leqslant n$, then we get $l \leq k_{n+1} \leq l+1$ since the width of the chain decomposition either increases by one or does not change. When it increases, i.e., when the vertex $n+1$ is a new chain $Z_{l+1} = \{n+1\}$, then there is no chain Z_j, $1 \leq j \leq l$, which can be extended to $n+1$. More formally,

$$k_{n+1} = l+1 \Leftrightarrow \forall j, 1 \leq j \leq l\text{: } (\max(Z_j), n+1) \notin E.$$

This leads to the transition probability

$$\begin{aligned}\Pr(k_{n+1} = l+1 \mid k_n = l) &= \Pr(\forall j, 1 \leq j \leq l\text{: } (\max(Z_j), n+1) \notin E) \\ &= (1-p)^l. \end{aligned} \tag{6}$$

Note that the l possible edges are independent. This implies

$$\Pr(k_{n+1} = l \mid k_n = l) = 1 - \Pr(k_{n+1} = l+1 \mid k_n = l) = 1-(1-p)^l. \tag{7}$$

Remark. Let $k_{n+1} = k_n$; then the current chain decomposition $Z_1, \ldots, Z_l$ can be extended to the new node $n+1$, and Algorithm B links the vertex $n+1$ to the chain with lowest index. Let $(\max(Z_i), n+1) \in E$ and

$$(\max(Z_1), n+1) \notin E, \ldots, (\max Z_{i-1}) \notin E;$$

Then Algorithm B connects $n+1$ with chain Z_i. It is easy to see that this special choice has no influence on the future growth of the chain decomposition.

In the same way as k we treat $\gamma_n^* = |\mathrm{out}_n^*(1)|$:

$$\begin{aligned}\Pr(\gamma_{n+1}^* = l+1 \mid \gamma_n^* = l) &= \Pr(\exists w \in \mathrm{out}_n^*(1)\text{: } (w, n+1) \in E) \\ &= 1 - \Pr(\forall w \in \mathrm{out}_n^*(1)\text{: } (w, n+1) \notin E) \\ &= 1-(1-p)^l \end{aligned} \tag{8}$$

and

$$\Pr(\gamma_{n+1}^*(1) = l \mid \gamma_n^* = l) = 1 - \Pr(\gamma_{n+1}^* = l+1 \mid \gamma_n^* = l) = (1-p)^l. \tag{9}$$

With the additional notation $k_{n,l} =_{\text{def}} \Pr(k_n = l)$ and $\gamma_{n,l}^* =_{\text{def}} \Pr(\gamma_n^* = l)$ we find that $k_{n,l}$, $\gamma_{n,l}^*$ satisfy the following recurrence.

Lemma 3.2. $\forall n \in \mathbb{N}, \forall l, 1 \leq l \leq n,$

$$k_{1,1} = 1, \qquad k_{n,l} = (1-(1-p)^l) \cdot k_{n-1,l} + (1-p)^{l-1} \cdot k_{n-1,l-1} \tag{10}$$

and

$$\gamma_{1,1}^* = 1, \qquad \gamma_{n,l}^* = (1-p)^l \cdot \gamma_{n-1,l}^* + (1-(1-p)^{l-1}) \cdot \gamma_{n-1,l-1}^*. \tag{11}$$

Proof. Clear by the preceding discussion. □

This description shows that k_n (γ_n^* respectively) is a discrete-time, pure-birth process (see [4]). By a discrete-time, pure-birth process we understand a sequence of random variables X_t, $t \in \mathbb{N}$, assuming the states $l = 1, 2, 3, \ldots$ with corresponding probabilities $P_{t,l}$ and a sequence of transition probabilities λ_l, $0 \leqslant \lambda_l \leqslant 1$ and $l \in \mathbb{N}$, so that

$$P_{1,1} = 1 \quad \text{and} \quad P_{t,l} = (1-\lambda_l) \cdot P_{t-1,l} + \lambda_{l-1} \cdot P_{t-1,l-1},$$

i.e., the process starts at epoch 1 from state 1; direct transitions from a state l are only possible to $l+1$; these transitions have probability λ_l. To provide for an easy treatment we first give a very useful identity of this kind of birth process.

Let φ be a real function with

$$\varphi(l) = \sum_{j=1}^{l-1} \frac{1}{\lambda_j} \quad \forall l \in \mathbb{N}.$$

Then we state the following lemma.

Lemma 3.3. $\mathrm{E}(\varphi(X_t)) = t-1$, $\forall t \in \mathbb{N}$.

Proof. We use induction on t.

$$(t=1): \quad \mathrm{E}(X_1) = \sum_{l=1}^{1} \varphi(l) \cdot P_{1,1} = 0 \cdot 1 = 0.$$

($t \geqslant 2$): Our induction hypothesis (I.H.) is $\mathrm{E}(\varphi(X_{t-1})) = t-2$. Then we have

$$\begin{aligned}
\mathrm{E}(\varphi(X_t)) &= \sum_{l=1}^{t} \varphi(l) \cdot P_{t,l} \\
&= \sum_{l=1}^{t} \varphi(l) \cdot ((1-\lambda_l) \cdot P_{t-1,l} + \lambda_{l-1} \cdot P_{t-1,l-1}) \\
&= \underbrace{\sum_{l=1}^{t} \varphi(l) \cdot P_{t-1,l}}_{= \mathrm{E}(\varphi(X_{t-1})) = t-2} + \sum_{l=1}^{t} \varphi(l) \cdot \lambda_{l-1} \cdot P_{t-1,l-1} - \sum_{l=1}^{t} \varphi(l) \cdot \lambda_l \cdot P_{t-1,l} \\
&= t-2 + \underbrace{\sum_{l=1}^{t} \varphi(l) \cdot \lambda_{l-1} \cdot P_{t-1,l-1}}_{= \sum_{l=0}^{t-1} \varphi(l+1) \cdot \lambda_l \cdot P_{t-1,l}} - \sum_{l=1}^{t} \varphi(l) \cdot \lambda_l \cdot P_{t-1,l}
\end{aligned}$$

(note that $P_{t-1,0} = 0 = P_{t-1,t}$)

$$= t-2 + \sum_{l=1}^{t-1} \varphi(l+1) \cdot \lambda_l \cdot P_{t-1,l} - \sum_{l=1}^{t-1} \varphi(l) \cdot \lambda_l \cdot P_{t-1,l}$$

(since $\varphi(l+1)=\varphi(l)+1/\lambda_l$)

$$= t-2+\sum_{l=1}^{t-1}\underbrace{(\varphi(l)\cdot\lambda_l\cdot P_{t-1,l}-\varphi(l)\cdot\lambda_l\cdot P_{t-1,l})}_{=0}+\frac{1}{\lambda_l}\cdot P_{t-1,l}$$

$$= t-2+\underbrace{\sum_{l=1}^{t-1}\frac{1}{\lambda_l}\cdot\lambda_l\cdot P_{t-1,l}}_{=1}$$

$$= t-2+1=t-1. \qquad \square$$

By use of Lemma 3.3 a simple deduction leads to Lemmas 3.4 and 3.5.

Lemma 3.4

$$\mathrm{E}(k(G_{n,p}))\leqslant\frac{\log(p\cdot(n-1)+1)}{-\log(1-p)}+1.$$

Proof. With expression (6) there is

$$\lambda_l(k)=\Pr(k_n=l+1\,|\,k_{n-1}=l)=(1-p)^l.$$

By a simple application of the summation formula for geometric series (see Appendix A) we get

$$\varphi(l)=\sum_{j=1}^{l-1}\frac{1}{\lambda_j(k)}=\sum_{j=1}^{l-1}\frac{1}{(1-p)^j}=\frac{1}{p}\cdot\left(\left(\frac{1}{1-p}\right)^{l-1}-1\right)\quad\forall l,\ 1\leqslant l\leqslant n.$$

Of course, $\varphi(l)$ is an exponential function and so $\varphi(l)$ is convex, i.e.,

$$\forall x_1,\ldots,x_n\in\mathbb{R}:\quad \alpha_1,\ldots,\alpha_n\in\mathbb{R}^+ \text{ with } \sum_{l=1}^{n}\alpha_l=1.$$

We get

$$\varphi(\alpha_1\cdot x_1+\cdots+\alpha_n\cdot x_n)\leqslant\alpha_1\cdot\varphi(x_1)+\cdots+\alpha_n\cdot\varphi(x_n).$$

This implies with $\alpha_l=k_{n,l}$ and $x_l=l$, $1\leqslant l\leqslant n$, Jensen's inequality

$$\varphi(\mathrm{E}(k_n))\overset{\text{def}}{=}\varphi\left(\sum_{l=1}^{n}l\cdot k_{n,l}\right)\leqslant\sum_{l=1}^{n}\varphi(l)\cdot k_{n,l}\overset{\text{def}}{=}\mathrm{E}(\varphi(k_n)).$$

Now we apply Lemma 3.3 and the inverse function φ^{-1} to φ given by

$$\varphi^{-1}(x)=\frac{\log(p\cdot x+1)}{-\log(1-p)}+1.$$

We find

$$\varphi(\mathrm{E}(k_n)) \leqslant \mathrm{E}(\varphi(k_n)) \overset{(\mathrm{L.3.3})}{=} n-1$$
$$\Leftrightarrow \varphi(\mathrm{E}(k_n)) \leqslant n-1$$
$$\Leftrightarrow \varphi^{-1}(\varphi(\mathrm{E}(k_n))) \leqslant \varphi^{-1}(n-1)$$
$$\Leftrightarrow \mathrm{E}(k_n) \leqslant \frac{\log(p\cdot(n-1)+1)}{-\log(1-p)}+1. \qquad \square$$

Lemma 3.5

$$\mathrm{E}(|\mathrm{out}^*(1)(G_{n,p})|) \geqslant n+1-\frac{|\log p|+1}{p}.$$

Proof. By term (7) we have $\lambda_l = \lambda_l(\gamma^*) = 1-(1-p)^l$ and, consequently, for $l \in \mathbb{N}$,

$$\varphi(l) = \sum_{j=1}^{l-1} \frac{1}{\lambda_j} = \frac{1}{p} + \sum_{j=2}^{l-1} \frac{1}{1-(1-p)^j}$$
$$\leqslant \frac{1}{p} + \sum_{j=2}^{l-1} \int_{j-1}^{j} \frac{dx}{1-(1-p)^x} = \frac{1}{p} + \int_{1}^{l-1} \frac{dx}{1-(1-p)^x}$$

(see [1, p. 87])

$$= \frac{1}{p} + (l-1) - \underbrace{\frac{\log(1-(1-p)^{l-1})}{\log(1-p)}}_{\geqslant 0} - 1 + \frac{\log(1-(q-p))}{\log(1-p)}$$
$$\leqslant (l-2) + \frac{|\log p|+1}{p}.$$

And, consequently, $\varphi(l) \leqslant (l-2) + (|\log p|+1)/p =_{\mathrm{def}} \phi(l)$. This implies

$$\mathrm{E}(\varphi(\gamma_n^*)) = \sum_{l=1}^{n} \varphi(l)\cdot\gamma_{n,l}^* \leqslant \sum_{l=1}^{n} \phi(l)\cdot\gamma_{n,l}^* = \mathrm{E}(\phi(\gamma_n^*))$$

also by Lemma 3.3 and, by the fact that ϕ is a straight line,

$$n-1 = \mathrm{E}(\varphi(\gamma_n^*)) \leqslant \mathrm{E}(\phi(\gamma_n^*)) = \phi(\mathrm{E}(\gamma_n^*)) = \mathrm{E}(\gamma_n^*) + \frac{|\log p|+1}{p} - 2$$
$$\Leftrightarrow \mathrm{E}(\gamma_n^*) \geqslant n+1-\frac{|\log p|+1}{p}. \qquad \square$$

Using the notation $\gamma_n^{\mathrm{red}} = \gamma^{\mathrm{red}}(1)(G_{n,p})$ for the size of the transitive reduction of the first vertex, we show that the lower bound for $\mathrm{E}(\gamma_n^*)$ gives an upper bound for the expected value of the reduction $\mathrm{E}(\gamma_n^{\mathrm{red}})$. This can be reached by comparing $\Delta\mathrm{E}(\gamma_n^*) = \Pr((1,n) \in E^*)$ with $\Delta\mathrm{E}(\gamma_n^{\mathrm{red}}) = \Pr((1,n) \in E_{\mathrm{red}})$.

Lemma 3.6

$$\mathrm{E}(\gamma_n^{\mathrm{red}}) = \frac{p}{1-p} \cdot (n - \mathrm{E}(\gamma_n^*)).$$

Proof. Let A be the event "$(1, n) \in E^*$" with

$$\Pr((1, n) \in E^*) = \Pr(\exists w \in \mathrm{out}_{n-1}^*(1): (w, n) \in E).$$

Then, by splitting A on γ_{n-1}^*, we have

$$\Pr((1, n) \in E^*) = \sum_{l=1}^{n-1} \underbrace{\Pr(A \mid \gamma_{n-1}^* = l)}_{=(1-(1-p)^l)} \cdot \underbrace{\Pr(\gamma_{n-1}^* = l)}_{=\gamma_{n-1,l}^*}$$

$$= \sum_{l=1}^{n-1} (1-(1-p)^l) \cdot \gamma_{n-1,l}^*. \tag{12}$$

Let $n \geqslant 2$; then we find

$$\Pr((1, n) \in E_{\mathrm{red}}) = \Pr((1, n) \in E) \cdot \Pr(\forall w \in \mathrm{out}_{n-1}^*(1), w \neq 1: (w, n) \notin E)$$

$$= p \cdot \sum_{l=1}^{n-1} (1-p)^{l-1} \cdot \gamma_{n-1,l}^*$$

(splitting by γ_{n-1}^*)

$$= \frac{p}{1-p} \cdot \sum_{l=1}^{n-1} (1 - 1 + (1-p)^l) \cdot \gamma_{n-1,l}^*$$

$$= \frac{p}{1-p} \cdot \left(1 - \sum_{l=1}^{n-1} (1-(1-p)^l) \cdot \gamma_{n-1,l}^*\right)$$

Thus we arrive at

$$\Pr((1, n) \in E_{\mathrm{red}}) \overset{(12)}{=} \frac{p}{1-p} \cdot (1 - \Pr((1, n) \in E^*)) \tag{13}$$

Now the proof is easily completed:

$$\mathrm{E}(\gamma_n^{\mathrm{red}}) = \sum_{j=2}^{n} \mathrm{E}((1,j) \in E_{\mathrm{red}}) = \sum_{j=2}^{n} \Pr((1,j) \in E_{\mathrm{red}})$$

$$\overset{(13)}{=} \sum_{j=2}^{n} \frac{p}{1-p} \cdot (1 - \Pr((1,j) \in E^*))$$

$$= \frac{p}{1-p} \cdot \left(n - 1 - \sum_{j=2}^{n} \Pr((1,j) \in E^*)\right)$$

$$= \frac{p}{1-p} \cdot (n - \mathrm{E}(\gamma_n^*)). \qquad \square$$

Corollary 3.7. $\mathrm{E}(e_{\mathrm{red}}) \leqslant n \cdot (|\log p| + 2)$.

Proof

$$\mathrm{E}(e_{\mathrm{red}}) = \mathrm{E}(\gamma^{\mathrm{red}}(1)) + \cdots + \mathrm{E}(\gamma^{\mathrm{red}}(n)) \leqslant n \cdot \mathrm{E}(\gamma^{\mathrm{red}}(1)) = n \cdot \mathrm{E}(\gamma_n^{\mathrm{red}}).$$

By Lemma 3.6,

$$\mathrm{E}(e^{\mathrm{red}}) = n \cdot \frac{p}{1-p} \cdot (n - \mathrm{E}(\gamma_n^*))$$

(by Lemma 5)

$$\leqslant n \cdot \frac{p}{1-p} \cdot \left(n - \left(n + 1 - \frac{|\log p| + 1}{p}\right)\right)$$

$$\leqslant n \cdot \left(\frac{|\log p|}{1-p} + 1\right)$$

(by Taylor)

$$\leqslant n \cdot \left(\frac{(1-p) + \frac{1}{2}(1-p)^2 + \frac{1}{3}(1-p)^3 + \cdots}{1-p} + 1\right)$$

$$\leqslant n \cdot (|\log p| + 2). \qquad \square$$

Corollary 3.8. *The expected running time of Algorithm* B, *according to Goralčíková and Koubek* [6], *is*

$$\mathrm{O}(\min(n^2 \cdot (|\log p + 2), p \cdot n^3)) = \mathrm{O}(n^2 \cdot \log n).$$

Proof. We have, by the preceding discussion,

Case $p > (\log n)/n$:

$$\mathrm{E}(n \cdot e_{\mathrm{red}}) \leqslant n^2 \cdot (|\log p| + 2) \leqslant \mathrm{O}(n^2 \cdot \log n).$$

Case $p \leqslant (\log n)/n$:

$$\mathrm{E}(n \cdot e_{\mathrm{red}}) \leqslant \mathrm{E}(n \cdot e) \leqslant p \cdot n^3 \leqslant \mathrm{O}(n^2 \cdot \log n). \qquad \square$$

In the remainder of this section we develop upper bounds for $\mathrm{E}(k \cdot e_{\mathrm{red}})$. Recall that $k \cdot e_{\mathrm{red}}$ is a product of two dependent random variables. In Lemma 3.12 we will understand e_{red} as the dependent variable, i.e., we use the interpretation

$$\mathrm{E}(k \cdot e_{\mathrm{red}}) = \mathrm{E}(k) \cdot \mathrm{E}(e_{\mathrm{red}} \,|\, k).$$

This analysis is prepared in the next three lemmas. First we determine the expected size of chain $Z = Z(G_{n,p})$ constructed in Algorithm B. The repeated application of Lemma 3.9 shows that

$$\mathrm{E}(|V_i|) = (1-p)^{i-1} \cdot n - h(\Pr(k(G_{n,p}) \geqslant i)).$$

By simple transformations we can see that function $f(i) = \Pr(k(G_{n,p}) \geqslant i)$ decreases exponentially.

Now we analyse the construction of the chain decomposition (Algorithm B) more exactly. The algorithm iteratively constructs a chain $Z = Z(G)$, deletes Z from G and starts again with graph $G' = G - Z$. Retain the notations:

$$G_1 = (V_1, E) = G_{n,p}, \qquad G_i = (V_i, E),$$
$$V_{i+1} = V_i - Z(G_i) = V_i - Z_i.$$

For the chain $Z = Z(G = (V, E))$ there is

(1) $\min(V) \in Z$, and

(2) $v \in Z \Rightarrow \min(\text{out}(v)) \in Z$.

Now we are going to compute the expected size of $Z(G_{n,p})$ in our random graph model.

Lemma 3.9. $E(|Z(G_{n,p})|) = 1 + p \cdot (|V| - 1)$.

Proof. Let $T \subseteq V = \{1, \ldots, n\}$ with $T = \{1 = t_1 < t_2 < \cdots < t_s\}$. Then $T = Z(G)$ if and only if

$$(1, h) \notin E, \forall 1 < h < t_2 \wedge (1, t_2) \in E \wedge (t_2, h) \notin E, \forall t_2 < h < t_3 \wedge \ldots$$
$$\wedge (t_s, h) \notin E, \forall t_s < h \leq n.$$

Since all these events are independent, we get

$$\Pr(T = Z(G)) = p^{s-1} \cdot (1-p)^{(n-1)-(s-1)} = p^{|T|-1} \cdot (1-p)^{n-|T|}.$$

For $T_1, T_2 \subseteq V$, we obtain

$$T_1 = Z(G) \wedge T_2 = Z(G) \Rightarrow T_1 = T_2$$

such that we have $\forall T_1, T_2 \subseteq V, T_1 \neq T_2$:

$$\Pr(T_1 = Z \vee T_2 = Z) = \Pr(T_1 = Z) + \Pr(T_2 = Z)$$

and further

$$\Pr(|Z(G)| = l + 1) = \binom{n-1}{l} \cdot p^l \cdot (1-p)^{(n-1)-l}.$$

But this is a binomial distribution (see [4, Vol. 1, p. 146]), which implies

$$E(|Z(G)|) = 1 + p \cdot (n-1). \qquad \square$$

Since removal of $Z_1, \ldots, Z_i$ turns a random graph into a random graph, we get, by Lemma 3.9, $E(|Z_i|) = 1 + p \cdot (j-1)$ on the condition that $|V_i| = j$. This implies the following lemma.

Lemma 3.10

$$E(|V_i|) = (1-p)^{i-1} \cdot n - \sum_{j=1}^{i-1} \Pr(k(G) \geq i - j) \cdot (1-p)^j.$$

Proof. We first show by induction on i that

$$E(|V_i|)=(1-p)^{i-1}\cdot n-\sum_{j=1}^{i-1}(1-p)^j+\sum_{j=1}^{i-1}\Pr(|V_j|=0)\cdot(1-p)^{i-j}.$$

$(i=1)$: $E(|V_1|)=|V|=n=p^0\cdot n-0.$

$(i\geq 2)$: $E(|V_{i+1}|)=E(|V_i|)-E(|Z_i|)$

$$=E(|V_i|)-\left(\sum_{j=0}^{n}E(|Z_i|\,|\,|V_i|=j)\cdot\Pr(|V_i|=j)\right)$$

$$\overset{(L.3.9)}{=}E(|V_i|)-\sum_{j=1}^{n}(1-p\cdot(j-1))\cdot\Pr(|V_i|=j)$$

$$=E(|V_i|)-\Bigg(\underbrace{\sum_{j=0}^{n}(1-p\cdot(j-1))\cdot\Pr(|V_i|=j)}_{=1-p+p\cdot\underbrace{\sum_{j=0}^{n}j\cdot\Pr(|V_i|=j)}_{=E(|V_i|)}}-(1-p)\cdot\Pr(|V_i|=0)\Bigg)$$

$$=E(|V_i|)-(p\cdot E(|V_i|)+(1-p)-(1-p)\cdot\Pr(|V_i|=0))$$

$$=(1-p)\cdot E(|V_i|)-(1-p)+(1-p)\cdot\Pr(|V_i|=0)$$

$$\overset{(I.H.)}{=}(1-p)\left((1-p)^{i-1}\cdot n-\sum_{j=1}^{i-1}(1-p)^j+\sum_{j=1}^{i-1}\Pr(|V_j|=0)(1-p)^{i-j}\right)$$

$$-(1-p)+(1-p)\cdot\Pr(|V_i|=0)$$

$$=(1-p)^i\cdot n-\sum_{j=1}^{i}(1-p)^j+\sum_{j=1}^{i}\Pr(|V_j|=0)\cdot(1-p)^{i+1-j}.$$

And this ends the induction proof of our first statement. This expression for $E(|V_i|)$ is equivalent to

$$E(|V_i|)=(1-p)^{i-1}\cdot n-\sum_{j=1}^{i-1}(1-p)^j+\underbrace{\sum_{j=1}^{i-1}\Pr(|V_j|=0)\cdot(1-p)^{i-j}}_{=\sum_{j=1}^{i-1}\Pr(|V_{i-j}|=0)\cdot(1-p)^j}$$

$$=(1-p)^{i-1}\cdot n-\sum_{j=1}^{i-1}\underbrace{(1-\Pr(|V_{i-j}|=0))}_{\substack{=\Pr(|V_{i-j}|\neq 0)\\=\Pr(k(G)\geq i-j)}}\cdot(1-p)^j$$

$$E(|V_i|)=(1-p)^{i-1}\cdot n-\sum_{j=1}^{i-1}\Pr(k(G)\geq i-j)\cdot(1-p)^j.\qquad\square$$

This last lemma allows us to give an upper bound for $\Pr(k(G_{n,p})\geq i)$.

Lemma 3.11. $\forall 1 \leqslant i \leqslant n$:

$$\Pr(k(G_{n,p}) \geqslant i) \leqslant (1-p)^{i-1} \cdot n.$$

Proof. Since $E(|V_i|) \geqslant 0$, we infer from Lemma 3.10

$$\sum_{j=1}^{i-1} \Pr(k(G) \geqslant i-j) \cdot (1-p)^j \leqslant (1-p)^{i-1} \cdot n.$$

Now, note that there is

$$\Pr(k(G) \geqslant i-1) \leqslant \Pr(k(G) \geqslant i-2) \leqslant \cdots \leqslant \Pr(k(G) \geqslant 1) = 1.$$

This implies

$$\Pr(k(G) \geqslant i-1) \cdot \sum_{j=1}^{i-1} (1-p)^j \leqslant (1-p)^{i-1} \cdot n$$

$$\Rightarrow \Pr(k(G) \geqslant i-1) \cdot (1-p) \cdot \left(\frac{1-(1-p)^i}{1-(1-p)}\right) \leqslant (1-p)^{i-1} \cdot n$$

$$\Rightarrow \Pr(k(G) \geqslant i-1) \leqslant (1-p)^{i-2} \cdot n. \qquad \square$$

Lemma 3.11 leads to the first upper bound for $E(k \cdot e_{\text{red}})$.

Lemma 3.12. *Let* $k_o =_{\text{def}} \lceil -(4 \cdot \log n)/\log(1-p) \rceil$. *Then*

$$E(k \cdot e_{\text{red}}) \leqslant k_0 \cdot E(e_{\text{red}}) + 1.$$

Proof. Let $k_0 = \lceil -(4 \cdot \log n)/\log(1-p) \rceil$; then we get by, Lemma 3.11,

$$\begin{aligned} \Pr(k(G) \geqslant k_0 + 1) &\leqslant (1-p)^{k_0} \cdot n \\ &\leqslant (1-p)^{-(4 \cdot \log n)/\log(1-p)} \cdot n \\ &\leqslant \frac{n}{\exp(4 \cdot \log n)} \leqslant \frac{1}{n^3}. \end{aligned} \tag{14}$$

This implies for our product $k \cdot e_{\text{red}}$:

$$\begin{aligned} E(k \cdot e_{\text{red}}) &= \sum_{l=1}^{n} l \cdot E(e_{\text{red}} | k(G) = l) \cdot \Pr(k(G) = l) \\ &= \sum_{l=1}^{k_0} l \cdot E(e_{\text{red}} | k(G) = l) \cdot \Pr(k(G) = l) \\ &\quad + \sum_{l=k_0+1}^{n} l \cdot E(e_{\text{red}} | k(G) = l) \cdot \Pr(k(G) = l) \\ &\leqslant k_0 \cdot \underbrace{\sum_{l=1}^{k_0} E(e_{\text{red}} | k(G) = l) \cdot \Pr(k(G) = l)}_{\leqslant E(e_{\text{red}})} + n^3 \cdot \underbrace{\sum_{l=k_0+1}^{n} \Pr(k(G) = l)}_{= \Pr(k(G) \geqslant k_0 + 1)} \end{aligned}$$

(by (14))

$$\leqslant k_0 \cdot E(e_{\text{red}}) + n^3 \cdot \frac{1}{n^3}. \qquad \square$$

We need a better bound for small p's.

Lemma 3.13. $E(k\cdot e_{red})\leq n\cdot E(|out(1)|)\cdot(E(k)+1)$.

Proof. First we need an upper bound for $E(k(G)\mid\gamma(1)=l)$.

We claim

$$E(k(G_{n,p})\mid\gamma(1)=l)\leq 1+E(k(G_{n,p})). \tag{15}$$

($l=0$): Then $Z_1(G_{n,p})=\{1\}$ and $G-\{1\}$ is a random graph with $n-1$ vertices. So we get

$$E(k(G_{n,p})\mid\gamma(1)=0)=1+E(k(G_{n-1,p}))\leq 1+E(k(G_{n,p})).$$

($l\geq 1$): Then $\gamma(1)\geq 1$ implies $|Z_1(G_{n,p})|\geq 2$ and $G-Z_1(G)$ is a random graph with $\leq n-2$ vertices. We have

$$E(k(G_{n,p})\mid\gamma(1)=l)\leq 1+E(k(G_{n-2,p}))\leq 1+E(k(G_{n,p})).$$

This implies for the product

$$E(k\cdot e_{red})\leq E(k\cdot e)=E(k\cdot\gamma(1))+\cdots+E(k\cdot\gamma(n))\leq n\cdot E(k\cdot\gamma(1))$$

which implies

$$E(k\cdot e_{red})\leq n\cdot\sum_{l=0}^{n-1} l\cdot E(k(G)\mid\gamma(1)=l)\cdot \Pr(\gamma(1)=l)$$

(by (15))

$$\leq n\cdot\sum_{l=0}^{n-1} l\cdot(E(k(G))+1)\cdot \Pr(\gamma(1)=l)$$

$$\leq n\cdot(E(k(G))+1)\cdot E(\gamma(1)). \qquad \square$$

4. Conclusion

We presented an improved algorithm for computing the transitive closure of an acyclic digraph with running time $O(k\cdot e_{red})$, where e_{red} is the number of edges in the transitive reduction and k is the width of the chain decomposition, a partition of V into distinct paths. To analyse the expected values of k, e_{red}, e^*, $k\cdot e_{red}$ we used the $G_{n,p}$ model of a random graph. We found

$$\Pr(k(G_{n,p})\geq i)\leq(1-p)^{i-1}\cdot n \quad \forall 1\leq i\leq n,$$

$$E(k)=O\left(\frac{\log(p\cdot n)}{p}\right), \qquad E(\gamma^*)=\Omega\left(n-\frac{\log p}{p}\right),$$

$$E(\gamma_n^{red})=\frac{p}{1-p}\cdot(n-E(\gamma_n^*)),$$

$$E(e_{red})=O(\min(n\cdot|\log p|, p\cdot n^2))=O(n\cdot\log n),$$

$$\mathrm{E}(k\cdot e_{\mathrm{red}})=\begin{cases}\mathrm{O}(n^2) & \text{for } \log^2 n/n\leq p<1,\\ \mathrm{O}(n^2\cdot\log\log n) & \text{otherwise.}\end{cases}$$

Our data structure for representing the transitive closure, namely niv(v) for a vertex v, the map id(v) and the chain decomposition $Z_1,\ldots,Z_k$ only used space $\mathrm{O}(n\cdot k)$ in contrast to $\mathrm{O}(e^*)$ of previous methods. Nevertheless we can execute the test "$w\in\mathrm{out}^*(v)$" in time O(1). Moreover, with this data structure it is easy to compute, e.g., $\mathrm{out}^*(v)\cap\mathrm{out}^*(v)$ in time $\mathrm{O}(k)$ and in general k is very much smaller than $|\mathrm{out}^*(w)|+|\mathrm{out}^*(v)|$

5. Further remarks

We think that the following questions are interesting:

- $\mathrm{E}(k\cdot e_{\mathrm{red}})\leq\mathrm{E}(k)\cdot\mathrm{E}(e_{\mathrm{red}})$? Conjecture: Yes.
- Do simple limit theorems exist for the k-, or γ^*-distribution?

Acknowledgment

This paper owes much to the help that I have received from Kurt Mehlhorn. My thanks to him for this support. Philippe Flajolet has published a paper on "Probabilitic counting" [5]. In this paper he has dealt with the k-distribution independently from our work. In particular, he has obtained the term in Lemma 3.4 and shown that the variance of the k-distribution is very small.

A. Appendix

Claim A.1

$$\sum_{j=1}^{l-1}\frac{1}{(1-p)^j}=\frac{1}{p}\cdot\left(\left(\frac{1}{1-p}\right)^{l-1}-1\right)\quad\forall l\in\mathbb{N}.$$

Proof

$$\sum_{j=1}^{l-1}\frac{1}{(1-p)^j}=\sum_{j=1}^{l-1}\left(\frac{1}{1-p}\right)^j=\left(\frac{1}{1-p}\right)\cdot\left(\sum_{j=0}^{l-2}\left(\frac{1}{1-p}\right)^j\right)$$
$$=\left(\frac{1}{1-p}\right)\cdot\left(\frac{1-(1-p)^{-(l-1)}}{1-(1-p)^{-1}}\right)=\frac{1}{p}\cdot\left(\left(\frac{1}{1-p}\right)^{l-1}-1\right).\qquad\square$$

Claim A.2

$$\varphi(x)=\frac{1}{p}\cdot\left(\left(\frac{1}{1-p}\right)^{x-1}-1\right)\;\Rightarrow\;\varphi^{-1}(x)=\frac{\log(p\cdot x-1)}{-\log(1-p)}+1.$$

Proof

$$\varphi(\varphi^{-1}(x)) = \frac{1}{p} \cdot \left(\left(\frac{1}{1-p} \right)^{(\log(p\cdot x+1)/(-\log(1-p))+1)-1} - 1 \right)$$

$$= \frac{1}{p} \cdot \left(\left(\frac{1}{1-p} \right)^{\log(p\cdot x+1)/(\log 1 - \log(1-p))} - 1 \right)$$

$$= \frac{1}{p} \cdot \left(\left(\left(\frac{1}{1-p} \right)^{1/-\log(1-p)} \right)^{\log(p\cdot x+1)} - 1 \right)$$

$$= \frac{1}{p} \cdot (\exp(\log(p\cdot x+1)) - 1)$$

$$= \frac{1}{p} \cdot (p\cdot x + 1 - 1) = x. \qquad \square$$

References

[1] I.N. Bronstein and K.A. Semendjajew, *Taschenbuch der Mathematik* (Deutsch, Frankfurt/Main, 1981).

[2] M. O'hEigeartaigh, J.K. Lenstra and A.H.G. Rinnooy Kan, *Combinatorial Optimization* (Wiley, New York, 1985).

[3] P. Erdös and J. Spencer, *Probabilistic Methods in Combinatorics* (Academic Press, New York, 1974).

[4] W. Feller, *An Introduction to Probability Theory and Its Applications, Vol. 1-2* (Wiley, New York, 1960 and 1966).

[5] Ph. Flajolet, Approximate counting: a detailed analysis, *BIT* **25** (1985) 113-134.

[6] A. Goralčíková and V. Koubek, A reduct and closure algorithm for graphs, in: *Proc. Conf. on Mathematical Foundations of Computer Science* (1979) Lecture Notes in Computer Science **74** (Springer, Berlin, 1979) 301-307.

[7] A.J. Jammel and H.G. Stiegler, On expected costs of deadlock detection, *Inform. Process. Lett.* **11** (1980) 229-231.

[8] K. Mehlhorn, *Data Structures and Algorithms, Vol. 2: Graph Algorithms and NP-Completeness,* EATCS Monographs in Computer Science **2** (Springer, Berlin, 1984).

[9] C.P. Schnorr, An algorithm for transitive closure with linear expected time, *SIAM J. Comput.* **7** (1978) 124-133.