RAIRO. INFORMATIQUE THÉORIQUE ET APPLICATIONS

PAOLA ALIMONTI
STEFANO LEONARDI
ALBERTO MARCHETTI-SPACCAMELA

Average case analysis of fully dynamic reachability for directed graphs

RAIRO. Informatique théorique et applications, tome 30, n° 4 (1996), p. 305-318

<http://www.numdam.org/item?id=ITA_1996__30_4_305_0>

© AFCET, 1996, tous droits réservés.

L'accès aux archives de la revue « RAIRO. Informatique théorique et applications » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM
Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 30, n° 4, 1996, pp. 305-318)

AVERAGE CASE ANALYSIS OF FULLY DYNAMIC REACHABILITY FOR DIRECTED GRAPHS † (*) (**)

by Paola ALIMONTI, Stefano LEONARDI and Alberto MARCHETTI-SPACCAMELA ([1])

Communicated by G. AUSIELLO

Abstract. – *We consider the problem of maintaining the transitive closure in a directed graph under edge insertions and deletions from the point of view of average case analysis. Say n the number of nodes and m the number of edges. We present a data structure that supports the report of a path between two nodes in $O(n \cdot \log n)$ expected time and $O(1)$ worst case time per update, and reachability queries in $O(1)$ expected time and $O(n \cdot \log n)$ expected amortized time per update. If $m > n^{4/3}$ then reachability queries can be performed in $O(1)$ expected time and $O(\log^3 n)$ expected amortized time per update. These bounds compare favorably with the best bounds known using worst case analysis. Furthermore we consider an intermediate model between worst case analysis and average case analysis: the semi-random adversary introduced in* [3].

Résumé. – *On considère le problème de préserver la fermeture transitive d'un graphe orienté pendant l'insertion et l'élimination d'arêtes du point de vue de l'analyse du cas moyen. Soit n le nombre de sommets et m le nombre d'arêtes. Nous introduisons une structure de donnée qui permet de supporter soit la recherche d'un chemin ente deux sommets en temps moyen $O(n \cdot \log n)$ et en temps $O(1)$ pour chaque mise à jour dans le cas pire, soit de requette concernent l'existence d'un chemin entre deux sommets en temps moyen $O(1)$ et en temps moyen amorti pour chaque mise à jour. Les bornes précédentes deviennent $O(1)$ et $O(\log^3 n)$ respectivement dans le cas où $m > n^{4/3}$. Les bornes obtenues sont meilleures vis-à-vis des bornes qu'on obtient dans l'analyse du pire cas. Enfin on considère un modèle d'analyse au milieu entre l'analyse du cas moyen et l'analyse du pire cas : l'adversaire « semi-random » introduit en* [3].

(*) Received September 1995.

(**) Work supported by: the ESPRIT Basic Research Action No. 7141 (ALCOM II); the Italian Project "Efficienza di Algoritmi e Progetto di Strutture Informative", Ministero dell'Università e della Ricerca Scientifica e Tecnologica; the Italian Project "Progetto Finalizzato Trasporti 2", Consiglio Nazionale delle Ricerche, Italy.

(†) Preliminary version of this work in [2].

([1]) Dipartimento di Informatica e Sistemistica, Università di Roma "la Sapienza", Italy.
E-mail: {alimon, leon, alberto}@athena.dis.uniromal.it

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/96/04/$ 7.00/© AFCET-Gauthier-Villars

1. INTRODUCTION

Significant progess has been recently made in the design of algorithms and data structures for dynamic graphs [1, 5, 6, 8, 11-13, 16-21, 24]. These data structures support insertions and deletions of edges and/or nodes in a graph, in addition to several types of queries. The goal is to compute the new solution in the modified graph without having to recompute it from scratch. Usually, the sequence of insertions/deletions of edges is not known in advance and each operation must be completed before the next operation is known. If the data structure supports only insertions or only deletions then it is said *partially dynamic,* while a data structure is said *fully dynamic* if it supports both insertions and deletions.

The problem of dynamic maintenance of edge and vertex connected components in undirected graphs has been widely studied. However for directed graphs, the problem of maintaining the transitive closure appears much more difficult than the problem of maintaining simple connectivity in undirected graphs. Let n be the number of vertices and m be the number of edges. Consider the partially dynamic problem in which insertion of edges and connectivity queries for undirected graphs, and reachability queries for directed graphs are allowed. The amortized time per operation is $O(n)$ for directed graphs, instead of $O(\alpha)$ for undirected graphs.

Then, for an arbitrary sequence of insertions and connectivity (for undirected graphs) or reachability (for directed graphs) queries between a pair of vertices, the update amortized time for directed graphs is $O(n)$ instead of $O(\alpha(n, n))$ for undirected graphs [16, 19, 23]. If we consider deletions of edges there are solutions for special classes of graphs such as directed acyclic graphs [17]. The fully dynamic problem has also been studied [11, 19, 21] but, to the best of our knowledge, no fully dynamic data structure exists for general directed graphs that, in the worst case, achieves a bound of $o(m)$ for reachability queries and update operations. Conversely, if we look to undirected graphs, the fully dynamic problem can be solved in $O(n^{1/2})$ per operation [7].

In this work we deal with the problem of on-line maintaining the transitive closure in a fully dynamic directed graph from the point of view of average case analysis rather than worst case analysis. This kind of analysis has been already applied to undirected graphs: in [20] a data structure that supports fully dynamic operations in $O(\log^3 n)$ amortized expected time is presented. The authors introduced the concept of stochastic graph process to model the evolution of a random graph under a sequence of random

insertions/deletions of edges. We extend their approach to directed graphs and we consider two different kinds of queries: *a*) *report-path* queries that report a path between two nodes if it exists; *b*) *reach* queries that report the information on reachability between two nodes.

We present algorithms and data structures that support the following operations:

- *report-path* queries in $O(n \cdot \log n)$ expected time and $O(1)$ worst case time per update;
- *reach* queries in a dense graph ($m > \Lambda \cdot n \cdot \log n$, with Λ constant) in $O(1)$ expected time and $O(n \cdot \log n)$ expected amortized time per update;
- *report-path* queries in $O(n^{1/2})$ expected time and $O(1)$ worst case time per update for a graph with $m > n^{3/2}$;
- *reach* queries in $O(1)$ expected time and $O(\log^3 n)$ expected amortized time per update, for a graph with $m > n^{4/3}$.

Our algorithms perform favorably with the best known algrithms for computing the transitive closure in random graphs [14, 15]. In fact Karp proposed an algorithm for computing the transitive closure of a sufficiently dense directed graph (with $m > \Lambda \cdot n \cdot \log n$, with Λ constant) in $O(n)$ expected time and linear space [14]. Notice that this algorithm answers only connectivity queries but does not allow the report of a path.

Finally, we consider an intermediate case between worst case analysis and average case analysis, the so-called semi-random graph model [3]. In this model the starting graph and the sequence of insertions and deletions is created by an adversary each of whose decisions is reversed with some small probability p. Our data structure supports report path queries in $O(n \cdot \log^2 n)$ for low reverse probability $p = n^{-1/2}$.

The paper is organized as follows. In Section 2 we introduce the model and we discuss some basic results from random graphs theory concerning the model. In Section 3 fully dynamic algorithms and data structures are given for report-path and reachability queries. In Section 4 the semi-random digraph model is defined and discussed. Section 5 outlines conclusions and presents open problems.

2. PRELIMINARIES

We first recall some standard notations. The model used for average case analysis is the standard random directed graph model [4]. Let $D = (V, E)$ be a directed graph with $n = |V|$ nodes and $m = |E|$ edges. Let $D_{n,m}$ be

the set of all the directed graphs with n nodes and m edges in which all the graphs have the same probability, and let $D_{n,p}$ be the set of all the directed graphs with n nodes, in which the edges are independently chosen to occur with probability p. In order to analyze randomly changing undirected graphs a *stochastic graph process* has been introduced [20]. We extend this notion to capture randomly changing directed graphs:

DEFINITION 2.1: *A* stochastic digraph process (sdp) *on a set of vertices* $V = \{1, 2, \ldots, n\}$ *is a Markov chain* $D^* = \{D_t\}_0^\infty$ *whose states are directed graphs on* V. *The process starts with* D_0 *being some directed graph on* V.

Let $D = (V, E_t)$ be the state of a sdp at time t; then we indicate by E_t^c all the possible edges not in E_t.

DEFINITION 2.2: *A stochastic digraph process on* $V = \{1, 2, \ldots, n\}$ *is called fair* (fsdp) *if*

1. D_0 *is the empty graph.*

2. *There is a* $t_1 > 0$ *such that* $\forall t \leq t_1$, D_t *is obtained from* D_{t-1} *by an addition of an edge uniformly at random among all edges in* E_t^c *(up to* t_1, *an edge is added at each* $t \leq t_1$).

3. *If* D_{t-1}, $t > t_1$, *is a clique (empty graph) then* D_t *is obtained from* D_{t-1} *by the deletion (addition) of a random edge; else*

4. D_t *is obtained from* D_{t-1} *by either an addition of one new edge which happens with probability* 1/2 *(and all not existing edges are equiprobable), or by the deletion of one existing edge which happens with probability* 1/2 *(and all existing edges are equiprobable).*

In the following we extend fundamental results of random graph theory to directed graphs.

LEMMA 2.1: *Let* D^* *be a fsdp on* $\{1, 2, \ldots, n\}$. *Let* $D_t = (V, E_t)$, *with* $|V| = n$ *and* $|E_t| = m$, *be the state of the process at time* t, *then* D_t *is a random digraph from* $D_{n,m}$.

Proof: By induction on the number of edges. D_1 is a random digraph from $D_{n,1}$, since it is obtained from the empty graph D_0 by randomly inserting an edge chosen with probability $(n \cdot (n-1))^{-1}$ among all possible edges. Assume that D_{t-1} is a random digraph obtained from $D_{n,t-1}$ through the insertion of an edge uniformly chosen at random among all edges in E_{t-1}^c. Since $|E_{t-1}^c| = n \cdot (n-1) - (t-1)$, every edge in E_{t-1}^c is chosen with

probability $(n \cdot (n-1) - (t-1))^{-1}$. Furthermore, any digraph with t edges can be obtained from t different graphs with $t-1$ edges. Since every digraph is equiprobable in D_{t-1}, the probability of a given digraph in D_t is

$$\binom{n \cdot (n-1)}{(t-1)}^{-1} \cdot \frac{t}{n \cdot (n-1) - (t-1)} = \binom{n \cdot (n-1)}{(t-1)}^{-1}$$

that is the probability of any digraph in $D_{n,t}$.

In the case of deletion, assume $| E_t | = m$ and $| E_{t-1} | = m+1$. We prove, again by induction that D_t is a random digraph from $D_{n,m}$. Suppose that D_{t-1} is a random digraph from $D_{n,m+1}$. A digraph with m edges can be obtained from $n \cdot (n-1) - m$ digraphs with $m+1$ edges by deleting one edge. Any digraph from $D_{n,m+1}$ has probability $\binom{n \cdot (n-1)}{m+1}^{-1}$ to occur, and every edge in a digraph from D_{t-1} is deleted with probability $(m+1)^{-1}$. Then, the probability of any digraph in D_t is

$$\binom{n \cdot (n-1)}{m+1}^{-1} \cdot \frac{(n \cdot (n-1) - m)}{m+1} = \binom{n \cdot (n-1)}{m}^{-1}$$

that is the probability of any digraph in $D_{n,m}$. □

In this paper we are mainly concerned with reachability, which is a monotone property (*i.e.* a property maintained while new edges are inserted). Next lemma extends to random directed graphs a result given in [4] for random graphs. The proof is omitted.

LEMMA 2.2: *If a monotone property holds for the model $D_{n,m}$ then it also holds for the corresponding model $D_{n,p}$, with $p = \frac{m}{n^2}$.*

The following lemma, by Karp [14], shows that certain known results about random graphs obtained through any standard sequential algorithms, such as breadth-first search or depth-first search, can be directly converted to results on random digraphs.

LEMMA 2.3: *Let G be drawn from $G_{n,p}$ and D be drawn from $D_{n,p}$. Then the random variables representing the number of vertices in the connected component of G containing vertex 1 and the number of vertices reachable from vertex 1 in D are identically distributed.*

Next result is a well known fundamental theorem proved by Erdös and Rényi [10] (*see* also [4]).

LEMMA 2.4: *Let $c \in R$ be fixed, and $p = \frac{\log+c+o(1)}{n}$. Then*

$$\text{Prob}\,(G_{n,p} \text{ is connected}) \to e^{-e^{-c}}.$$

If c is not constant then from [4] the following bound is derived.

LEMMA 2.5: *Let $p = \Lambda \cdot \frac{\log n}{n}$ and $\Lambda > 1$ is an appropriate constant. Then*

$$\text{Prob}\,(G_{n,p} \text{ is not connected}) < n^{-\Lambda+1}.$$

Proof: Let $p = \frac{\log n+c+o(1)}{n}$ with $c = (\Lambda-1)\cdot\log n$. Then, from lemma 2.4,

$$\text{Prob}\,(G_{n,p} \text{ is connected}) > e^{-e^{-(\Lambda-1)}} = e^{-e^{-(\Lambda-1)\log n}} \geq 1 - n^{-(\Lambda-1)}.$$

Hence the probability that Prob $(G_{n,p}$ is not connected$) < n^{-(\Lambda-1)}$. □

Using the previous lemma, we are able to extend the result on threshold connectivity from random graphs to random directed graphs.

LEMMA 2.6: *Let D be drawn from $D_{n,p}$. If $p > \Lambda \cdot \frac{\log n}{n}$, with $\Lambda > 2$ appropriate constant, then the digraph D is almost surely strongly connected with probability greater than $1 - n^{-\Lambda+2}$.*

Proof: Let $p = \Lambda \cdot \frac{\log n}{n}$. By lemmas 2.3 and 2.5 we have:

Prob (vertex 1 does not reach all vertices in $D_{n,p}$)

$$= \text{Prob}\,(G_{n,p} \text{ is not connected}) < n^{-\Lambda+1}$$

Therefore

Prob ($D_{n,p}$ is strongly connected)

$$= 1 - \text{Prob}\,(\exists \text{ a vertex that does not reach all vertices}) > 1 - n^{-\Lambda+2}.$$ □

Finally, observe that the following property holds for the expected degree of a node: the expected number of edges leaving or entering a node is $\frac{m}{n}$.

3. FULLY DYNAMIC DIRECTED GRAPHS CONNECTIVITY

In this paper we consider two kinds of operations on directed graphs: updates and queries. Namely we have:

- *report-path* (i, j): returns a path from node i to node j if such a path exists;
- *reach* (i, j): returns the information on reachability from node i to node j;

- *insert*: inserts a random edge;
- *delete*: deletes a random edge.

Now we introduce a data structure that allows to perform efficient reachability queries while edges are inserted and deleted. The underlying idea is to perform reachability queries in a subgraph randomly drawn from the original graph. Namely, we randomly extract a subgraph $D_K = (V_K, E_K)$ from the digraph $D = (V, E)$, where:

- V_K is a set of $k = \frac{c}{p} \cdot \log n$ independent nodes $V_K = \{v_1, \ldots, v_k\}$, randomly chosen from $V = \{1, \ldots, n\}$, where $p = \frac{m}{n^2}$ and c is an appropriate constant (*see* Section 3.1);
- $E_K = \{(g, h) \in E : g, h \in V_K\}$.

We refer to the graph $D_K = (V_K, E_K)$ as the *black graph*, and we will use it in place of the original graph D to perform connectivity queries. Furthermore we call the set of vertices V_K, the set of edges E_K, and the set of remaining vertices $V - V_K$, the sets of *black nodes, black edges,* and *white nodes* respectively.

Finally we associate to each white node $i \in V - V_K$:

- a double linked list $L_i = \{(i, g) \in E - E_K \mid g \in V_K\}$ of outgoing edges whose ending node is black;
- a double linked list $E_i = \{(g, i) \in E - E_K \mid g \in V_K\}$ of entering edges whose starting node is black.

With each edge we associate a pointer to the relative position in a list.

The time complexity of building the data structure is $O(m)$, and it takes $O(m)$ space.

In the following of this section we will study the expected time complexity of a fully dynamic algorithm for report-path queries, and the expected time complexity of a fully dynamic algorithm for reach queries.

3.1. Report-path query

The algorithm for report-path query first looks for a path formed only by edges in the black graph D_K possibly except the first and the last edge.

Let us first consider the case in which i and j are white nodes. If both L_i and E_j are not empty, then let (i, g) be an edge in L_i and (h, j) be an edge in E_j: the algorithm searches for a path P from g to h in D_K, and, if P exists, it returns the path $\langle (i, g), P, (h, j) \rangle$. If either L_i is empty, or E_j is empty, or there is no path from g to h in D_K, then the algorithm

searches the whole graph for a path from i to j. We will show that the last case happens with very low probability. The other cases are analogous with the following exceptions: if i is black then a path in D_K starting from i is searched; if j is black then a path in D_K arriving in j is searched.

To study the expected computational cost for report-path query we need the following preliminary lemmas.

LEMMA 3.1: *Let i be a white node. The probability that either E_i or L_i is empty is smaller than $2 \cdot n^{-c}$.*

Proof: Consider the set E_i. The probability that E_i is empty is equal to the probability that does not exist any edge from i to a black node, that is

$$(1-p)^k \leq e^{-k \cdot p}.$$

Since $k = \frac{c}{p} \cdot \log n$, follows that E_i is empty with probability smaller than $e^{-c \cdot \log n} = n^{-c}$. The same holds for L_i. □

Furthermore, we must ensure that with high probability there exists a path between any pair of black nodes.

LEMMA 3.2: *If $c > \Lambda$ then the graph D_K is strongly connected with probability greater than $1 - n^{-\Lambda+2}$.*

Proof: By lemma 2.6 it is sufficient to show that p_K, the edge probability of D_K satisfies $p_K > \Lambda \cdot \frac{\log k}{k}$, where it is easy to show that $p_K = p$, the edge probability in D. The inequality is satisfied for $k > \Lambda \cdot \frac{\log k}{p}$. Since we choose $k = \frac{c}{p} \cdot \log n$, we must take $c > \Lambda$. □

Then, the following lemma gives the computational cost for report-path query.

LEMMA 3.3: *If $c > \Lambda \geq 4$ then the expected cost for report-path query in a directed graph with $m > \Lambda \cdot n \cdot \log n$ edges is $O\left(\frac{n^2}{m} \cdot \log^2 n\right)$.*

Proof: We say A the event "the algorithm searches the whole graph". The expected cost x of a query can be bounded as follows:

$$E[x] \leq E[x \mid \neg A] + E[x \mid A] \cdot \text{Prob}(A).$$

$E[x \mid \neg A]$ is $O\left(\frac{n^2}{m} \cdot (\log^2 n)\right)$. In fact in this case the running time of the algorithm is dominated by the expected number of edges in D_K. The average

degree of a node in D_K is $p \cdot n \cdot \frac{k}{n} = p \cdot k$. Hence, the average number of edges in D_K is $O(k^2 \cdot p)$, that is $O\left(\frac{n^2}{m} \cdot (\log^2 n)\right)$, since $p = \frac{m}{n^2}$.

On the other side by lemmas 3.1 and 3.2 $\text{Prob}(A) < n^{-\Lambda+2} + 2 \cdot n^{-c}$; moreover in this case the running time of the algorithm is $O(m)$. Hence if $c > \Lambda \geq 4$ we have $E[x \mid A] \cdot \text{Prob}(A) = O(1)$. □

The data structure must be updated while edges are inserted and deleted. If an edge whose extreme nodes are both black is inserted or deleted then the edge is inserted or deleted in the black graph D_K. If and edge from a white note i to a black node is inserted then the edge is inserted in L_i, vice-versa if the edge is deleted, it is deleted from L_i. Analogously for an insertion or a deletion of an edge from a black node to a white node. All these updates are performed in constant time since we use doubled linked list and we maintain a pointer from any edge to its position in a list.

Furthermore, the structure must be rebuilt when the number of edges is either excessively increased or decreased. In the former case, by choosing a smaller number of black nodes, a lower query time is possible, while in the latter case the structure no longer satisfies the requirements stated in Lemmas 3.1 and 3.2.

Now let m^* be the number of edges in the graph when the structure is built. The same structure is used until m is within a range between $\frac{m^*}{2}$ and $2 \cdot m^*$. To satisfy the condition stated by Lemma 3.1, we take $k = 2 \cdot \frac{c}{p} \cdot \log n$, and we rebuild the data structure from scratch when m is outside the range. Since the structure is used for at least $O(m)$ updates, and the cost to build the data structure is $O(m)$, the amortized cost per update is $O(1)$. This amortized bound can be made worst case by using the following device. When the data structure for a given m is used for the first time, we also start to construct the data structure for both $\frac{m}{2}$ and $2\,m$. This is done only a part for each step, and reflecting insertions/deletions of edges. When the number of edges reaches either $\frac{m}{2}$ or $2\,m$, the appropriate data structure has been already built and can be used. The other one is now discarded.

Then we can state the following theorem:

THEOREM 3.4: *There exists a data structure such that the expected cost for report-path query in a directed graph with* $m > \Lambda \cdot n \cdot \log n$ *is* $O(n \cdot \log n)$, *and that can be updated in* $O(1)$ *worst case time for each insertion or deletion.*

Since our analysis applies only to graphs with number of edges greater than $\Lambda \cdot n \cdot \log n$, the expected computational cost for report-path query is

$O(n \cdot \log n)$. Clearly, also for $m \leq \Lambda \cdot n \cdot \log n$ the computational cost for a report-path query is $O(n \cdot \log n)$.

COROLLARY 3.5: *There exists a data structure such that the expected cost for report-path query in a directed graph is* $O(n \cdot \log n)$, *and that can be updated in* $O(1)$ *worst case time for each insertion or deletion.*

Notice that for a directed graph with number of edges $m > n^{3/2}$ the expected computational cost for report-path query is $O(n^{1/2})$.

3.2. Connect query

In the following of this section we present an algorithm to perform reach queries between any pair of nodes in a dense directed graph (*i.e.* $m > \Lambda \cdot n \cdot \log n$). In this case the report of a path that connects the pair of nodes is no longer required.

Our algorithm is based on an algorithm performing the transitive closure in dense directed graphs in $O(n)$ expected time, given by Karp [14]. The transitive closure is represented in a compact form with $O(n)$ expected space requirement. This data structure allows reach queries in $O(1)$ expected time, but, it does not support report-path queries.

The data structure is a slight modification of the data structure for report-path query; namely by representing in compact from the transitive closure of the black graph D_K. The algorithm proposed for report-path queries is then modified for reach queries in this way: instead of visiting D_K, check in its transitive closure, with $O(1)$ expected time. The query reach (i, j) returns *yes* if there exists a path from node i to node j in D_K possibly except the first and the last edge, otherwise it searches the whole graph D.

As we have shown in the previous section this is very unlikely to happen if c and Λ are large enough.

A straightforward analysis of this algorithm gives the following result:

LEMMA 3.6: *If* $c > \Lambda > 4$ *then the expected cost for reach query in a directed graph with* $m > \Lambda \cdot n \cdot \log n$ *is* $O(1)$.

Next, we consider the expected cost for updating the data structure when edges are inserted or deleted. The main problem concerns updates that modify the black graph D_K. In fact the transitive closure of the black graph must be updated and we are forced to rebuild it from scratch.

LEMMA 3.7: *The amortized expected cost to update the data structure for each insertion or deletion is* $O\left(\frac{n^4}{m^5} \cdot \log^3 n\right)$.

Proof: Since the probability that a random edge is black is $\frac{k^2}{n^2}$, and the expected cost for computing the transitive closure is $O(k)$, then the expected computational cost for computing from scratch the transitive closure after any insertion or deletion is $O\left(\frac{k^3}{n^2}\right) = O\left(\frac{n^4}{m^3} \cdot \log^3 n\right)$. Moreover the amortized cost per update is $O(1)$ since the data structure is used for at least $O(m)$ insertions or deletions and the cost to re-build it from scratch is $O(m)$.

LEMMA 3.8: *There exists a data structure such that the expected cost per reach query in a directed graph with* $m > \Lambda \cdot n \cdot \log n$ *is* $O(1)$, *and that can be update in* $O(n \cdot \log n)$ *amortized expected time for each insertion or deletion.*

For a directed graph with $m \geq n^{4/3}$ we state the following corollary.

COROLLARY 3.9: *There exists a data structure such that the expected cost per reach query in a directed graph with* $m \geq n^{4/3}$ *edges is* $O(1)$, *and that can be update in* $O(\log^3 n)$ *amortized expected time for each insertion or deletion.*

4. SEMI-RANDOM DIRECTED GRAPHS

In this section we consider the problem of performing efficient connectivity queries in a semi-random graph. In this case we consider an intermediate model between worst case analysis and average case analysis.

In the semi-random graph model each decision about the graph is not taken by a worst case adversary, but by an adversary each of whose decisions is reversed with small probability p. This type of adversary is derived from the *semi-random* source of Santha and Vazirani [22] and has been applied to approximate coloring of 3-chromatic graphs by Avrim Blum [3].

The decisions of the adversary regarding whether or not to include an edge in the starting graph, and the inclusion of any insertion or deletion of an edge in the sequence of updates, are accepted with probability $1 - p$ and rejected with probability p by the algorithm.

Notice that if the adversary would propose the repeated deletion of an edge, after a certain number of trials the probability that an edge is in the graph would be negligible. Then, the deletion (insertion) of an edge can be repeated only after the re-insertion (re-deletion) of the same edge. The insertion of an edge already existing or the deletion of an edge not existing does not modify the graph at all.

The semi-random graph generated by the previous procedure has the following property:

LEMMA 4.1: *Each edge has probability greater than $p - p^2$ to appear in a graph created by an adversary each of whose decisions is reversed with probability p.*

Proof: The property holds for the starting graph since each edge included by the adversary is in the graph with probability $1 - p$, and each edge not included by the adversary is in the graph with probability p. Moreover, insertions and deletions of the same edge are alternate in the sequence of updates given by the adversary. The probability that an edge appears in the graph after the i^{th} deletion is $P_i = p \cdot ((1-p) + p \cdot P_{i-1})$, where $p \cdot (1-p)$ is the probability that an edge is in the graph after the i^{th} deletion if the $i-1^{th}$ deletion successes, while $p^2 \cdot P_{i-1}$ is the probability that the edge is in the graph after the i^{th} deletion if the $(i-1)^{th}$ insertion fails. $P_1 = p \cdot (1-p)$ if the edge is in the starting graph, and $P_1 = p$ if the edge is not in the starting graph assuming that the adversary has imposed a deletion to exclude the edge from the starting graph. It can be shown by induction that the probability that each edge appears in the graph is p for an edge not included in the starting graph and $p \cdot (1 - p^i)$ with $i \geq 1$ for an edge included in the starting graph.

The above lemma states that we can always consider the presence of a random directed graph $D_{n,p-p^2}$ overlapping the semi-random graph.

The presence of a random directed graph hidden in the semi-random directed graph can be used to apply the same data structure developed in Section 3.1 for report-path queries. In particular we randomly choose $k = \frac{c}{p-p^2} \cdot \log n$ nodes to build the data structure where the constant c is large enough to satisfy lemmas 3.1 and 3.2. Observe that the number of edges in a semi-random directed graph is $O\left(p \cdot n^2\right)$ with exponentially low probability.

Report-path queries in semi-random directed graphs are performed by the same algorithm given in Section 3.1, and the expected computational cost of a query is given by the number of edges in the black graph, that is $O\left(k^2\right)$ in the worst case.

THEOREM 4.2: *There exists a data structure such that the expected computational cost for report-path query in a semi-random directed graph is* $O\left(\frac{1}{p^2} \cdot \log^2 n\right)$.

Notice that for low reverse probability such as $p = n^{-\frac{1}{2}}$ [3] the expected computational cost for report-path query is $O(n \cdot \log^2 n)$.

With regard to the update of the structure, observe that this is built only once since the set of k nodes randomly chosen depends only on the initial choice of p. Then the cost for building the structure can be considered as preprocessing, while the structure can be update in $O(1)$ worst case time bound for each insertion or deletion.

The data structure for reach queries given in Section 3.2 does not easily extend to semi-random directed graphs since the algorithm given in [14] for computing the transitive closure in random directed graphs does not apply to semi-random directed graphs.

5. CONCLUSIONS AND OPEN PROBLEMS

In this paper we have shown efficient solutions for answering reachability queries in stochastic random digraphs. The obvious open problem is to reduce the time requirements for reachability queries in the case of graphs with $m \leq n^{4/3}$. In particular, it would be interesting to show whether reachability in a stochastic digraph process can be dynamically solved in polylogarithmic time.

Another interesting open problem is to investigate the complexity of the shortest path problem in a stochastic digraph process.

ACKNOWLEDGMENTS

We thank an anonymous referee for useful suggestions.

REFERENCES

1. G. Ausiello, G. F. Italiano, A. Marchetti-Spaccamela and U. Nanni, Incremental algorithms for minimal length paths, *J. of Algorithms,* 1991, *12*, pp. 615-638.
2. P. Alimonti, S. Leonardi, A. Marchetti-Spaccamela and X. Messeguer, Average Case Analysis of Fully Dynamic Connectivity for Directed Graphs, *Proc. 19th Workshop on Graph-Theoretic Concepts in Computer Science,* LNCS, Springer-Verlag, 1993.
3. A. Blum, Some tools for approximate 3-coloring, *Proc. 31st IEEE Symp. on Foundations of Computer Science,* 1990.
4. B. Bollobas, *Random graphs,* Academic Press, 1985.
5. G. Di Battista and R. Tamassia, Incremental planarity testing", *Proc. 30th Annual Symp. on Fundations of Computer Science,* 1989.

6. G. Di Battista and R. Tamassia, On-line graph algorithms with SPQR-trees, *Proc. 17th Int. Coll. on Automata, Languages and Programming,* LNCS, Springer-Verlag, 1990.
7. D. Eppstein, Z. Galil and G. F. Italiano, Improved Sparsification, *Technical Report,* 93-20, Department of Information and Computer Science, University of California, Irvine, 1993.
8. D. Eppstein, Z. Galil, G. F. Italiano and A. Nissenzweig, Sparsification – A technique for speeding up dynamic graph algorithms, *Proc. 33rd Annual Symp. on Foundations of Computer Science,* 1992.
9. D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook and M. Young, Maintenance of a minimum spanning forest in a dynamic planar graph, *Proc. 1st ACM-SIAM Symp. on Discrete Algorithms,* S. Francisco, 1990.
10. P. Erdös and A. Rènyi, On random graphs I, *Publ. Math. Debrecen, 6,* pp. 290-297.
11. S. Even and H. Gazit, Updating distances in dynamic graphs, *Methods of Operations Research, 49,* 1985.
12. Z. Galil and G. F. Italiano, Fully dynamic algorithms for edge-connectivity problems, *Proc. 23rd ACM Symp. on Theory of Comp.,* 1991, pp. 317-327.
13. Z. Galil and G. F. Italiano, Reducing edge connectivity to vertex connectivity, *SIGACT News,* 1991, *22* (1), pp. 57-61.
14. R. M. Karp, The transitive closure of a random digraph, *Technical Report 89-047,* International Computer Science Institutive (ICSI), August 1989.
15. R. M. Karp and R. E. Tarjan, Linear expected-time algorithms for connectivity problems, *Proc. of the 11th. annual ACM Symp. on Theory of Computing,* 1980, pp. 368-377.
16. G. F. Italiano, Amortized efficiency of a path retriaval data structure, *Theoret. Comp. Sci.,* 1986, *48,* pp. 273-281.
17. G. F. Italiano, Finding paths and deleting edges in directed acyclic graphs, *Inf. Proc. Lett., 28,* 1988, pp. 5-11.
18. J. A. La Poutré, Maintenance of triconnected components of graphs, *Proc. 19th Int. Coll. Automata Languages and Programming,* Lect. Not. in Computer Sci., Springer-Verlag, 1992, pp. 354-365.
19. J. A. La Poutré and J. van Leeuwen, Maintenance of transitive closure and transitive reduction of graphs, *Proc. Work. on Graph Theoretic concepts in Comp. Sci.,* LNCS 314 Springer-Verlag, Berlin, 1985, pp. 106-120.
20. J. H. Reif, P. G. Spirakis and M. Yung, Re-randomization and average case analysis of fully dynamic graph algorithms, Alcom Technical Report ALCOM-II-054, 1994.
21. H. Rohnert, A dynamization of the all-pairs least cost path problem, *Proc. of the 2nd Symp. on Theoretical Aspects of Computer Science,* LNCS 182, Springer-Verlag, 1990.
22. M. Santha and U. V. Vazirani, Generating quasi-random sequences from semi-random sources, *Journal of Computer and Systems Science,* 1986, *33,* pp. 75-87.
23. R. E. Tarjan and Jan van Leeuwen, Worst case analysis of set union algorithms, *Journal of Assoc. Comput. Mach.,* 1984, *31,* pp. 245-281.
24. J. Westbrook, Algorithms and data structures for dynamic graph problems, Ph. D. Dissertation, *Tech. Rep.,* CS-TR-229–89, Dept. Of Computer Science, Princeton University, 1989.