SIAM J. COMPUT.
Vol. 16, No. 3, June 1987

EXPECTED COMPUTATION TIME FOR HAMILTONIAN PATH PROBLEM*

YURI GUREVICH† AND SAHARON SHELAH‡

Abstract. One way to cope with an NP-hard problem is to find an algorithm that is fact on average with respect to a natural probability distribution on inputs. We consider from that point of view the Hamiltonian Path Problem. Our algorithm for the Hamiltonian Path Problem constructs or establishes the nonexistence of a Hamiltonian path. For a fixed probability p, the expected run-time of our algorithm on a random graph with n vertices and the edge probability p is $O(n)$. The algorithm is adaptable to directed graphs.

Key words. average case complexity, NP-hard Hamiltonian circuit, Hamiltonian path, probability, random graphs, expected polynomial time, expected sublinear time

AMS(MOS) subject classifications. 05G35, 60C05, 68A10, 68A20

Introduction. One way to cope with an NP-hard decision problem $\exists y R(x, y)$ is to find an algorithm that is fast on average with respect to a natural probability distribution on inputs. See in this connection Levin (1984) and Johnson (1984). A similar approach can be taken if one is interested in exhibiting an object y that witnesses $R(x, y)$, and not only in the existence of a witness. We consider from this point of view the following version of the Hamiltonian Path Problem: given a graph with distinguished vertices Start and Finish, construct or establish the nonexistence of a Hamiltonian path from Start to Finish. A Hamiltonian path from Start to Finish can be defined as a linear ordering of all vertices such that Start is the first, Finish is the last, and every pair of neighbors forms an edge. Without loss of generality, we restrict our attention to the case when the vertices of the given graph form an initial segment $\{0, 1, \cdots, n-1\}$ of natural numbers, Start $= 0$ and Finish $= n-1$.

Let $\Gamma(n, p)$ be the popular probability space of graphs with vertices $0, 1, \cdots, n-1$ and the edge probability p (Erdös and Spencer (1974)). The probability, assigned to a particular graph G with vertices $0, 1, \cdots, n-1$, is the probability that G is the outcome of the following experiment: For each pair $\{i, j\}$ of distinct vertices, toss a coin with the probability p of the head; declare $\{i, j\}$ an edge if and only if the head turns up. Let $G_{n,p}$ be a random member of $\Gamma(n, p)$.

In order to deal with instances of an arbitrary size, consider a function $p(n)$ from positive integers to the real interval $[0, 1]$. The spaces $\Gamma(n, p(n))$ can be combined into one probability space S with respect to a probability distribution δ on positive integers (the meaning of $\delta(n)$ is the probability that a random graph has exactly n vertices); however, this may not be necessary for the following reason. The expected run-time

* Received by the editors June 27, 1985; accepted for publication July 26, 1986.

† Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, Michigan 48109-2122. The work of this author was supported in part by National Science Foundation grants MCS 83-01022 and DCR 85-03275.

‡ Institute of Mathematics and Computer Science, The Hebrew University, 91904 Jerusalem, Israel. The work of this author was supported in part by National Science Foundation grant MCS 81-01560. This paper was written in the fall of 1984 when this author was a visiting professor in both the Department of Electrical Engineering and Computer Science and the Department of Mathematics, The University of Michigan, Ann Arbor, Michigan 48109-2122.

of an algorithm on a random member of S with n vertices is bounded by some function $t(n)$ if the expected run-time of the algorithm on each $G_{n,p(n)}$ is bounded by $t(n)$.

There are several algorithms for the Hamiltonian Path Problem in the literature that almost always (with probability 1 minus the reciprocal of a polynomial in n) work fast on any $G_{n,p(n)}$ where $p(n)$ is sufficiently large to ensure a high probability of the existence of a Hamiltonian path. Angluin and Valiant (1979) survey the previous algorithms of that sort and give an elegant faster algorithm of their own; almost always, their algorithm runs in time $O(n(\log n)^2)$ on their Random Access Computer. Shamir (1983) gives a further result of that sort; his algorithm is slower but the requirement on p is weaker: $p(n) \geqq n^{-1}(\log_e n + c \log_e \log_e n)$ where $c > 3$. Bollobas, Fenner and Frieze (1985) weakened the requirement on $p(n)$ to the limit; almost always, their algorithm runs in time $O(n^{4+\varepsilon})$. The expected run-time of all these algorithms is exponential with one exception, which is the Bollobas, Fenner and Frieze algorithm in the case $p(n) \geqq \frac{1}{2}$. (In the technical report of Gurevich and Shelah (1984) we wrote that, to the best of our knowledge, there are no results in the literature on expected run-time of algorithms for the Hamiltonian Path Problem. The Bollobas, Fenner and Frieze paper appeared soon after that.)

We present here our algorithm HPA (Hamiltonian Path Algorithm) for the Hamiltonian Path Problem and estimate the run-time of HPA on $G_{n,p}$ in the case when p is fixed and n is varied. The algorithm constructs or establishes the nonexistence of a Hamiltonian path from Start to Finish. The exact form of input is not important; for the sake of definiteness, we suppose that graphs are represented by the adjacency matrices. It is important that a random access computer is used. It could be, for example, the randomized Random Access Computer of Angluin and Valiant (1979).

THEOREM 1. *There is a positive real number c satisfying the following condition. For any fixed probability p, the expected run-time of* HPA *on* $G_{n,p}$ *is* $cn/p + o(n)$.

COROLLARY. *For every p there is a constant d such that the expected run-time of HPA on* $G_{n,p}$ *is bounded by* dn/p.

The algorithm HPA is composed of three algorithms HPA1, HPA2 and HPA3. First HPA1 is applied. If HPA1 fails to solve the given instance of the Hamiltonian Path Problem, then HPA2 is applied. If HPA2 fails, then HPA3 is applied; HPA3 always succeeds. Let us comment briefly on each of the three algorithms.

The algorithm HPA1 is described in § 1. It tries to construct a Hamiltonian path, and definitely fails if the desired path does not exist. For a fixed p, the expected run-time of HPA1 on $G_{n,p}$ is $cn/p + o(n)$ where c is an absolute constant. The constant c is quite a modest number; its exact value depends on the machine model. For a fixed p, the probability that HPA1 fails on $G_{n,p}$ is $2^{**}(-O(\sqrt{n}))$.

The algorithm HPA2 is described in § 2. It tries to construct or establish the nonexistence of a Hamiltonian path. The idea is to take care of the troublesome vertices (those with relatively small degree) first. The run-time of HPA2 on a graph with n vertices is $O(n^2)$. For a fixed p, the probability that HPA2 fails on HPA1($G_{n,p}$), assuming that HPA1 has failed on $G_{n,p}$, is $2^{-3n+o(n)}$ where HPA1($G_{n,p}$) is the output of HPA1 on $G_{n,p}$ in the case that HPA1 fails on $G_{n,p}$.

The fastest (in the worst case) known algorithm for the Hamiltonian Path Problem is the dynamic programming algorithm (Bellman (1960), Held and Karp (1962)) which solves the Hamiltonian Path Problem in time $2^n \cdot n^{O(1)}$ where n is the number of vertices. We could adapt it as our third algorithm HPA3. A negative feature of the dynamic programming algorithm is that it requires exponential space. Our third algorithm HPA3, described in § 3, solves the Hamiltonian Path Problem for graphs with n vertices in time $2^{2n} \times 2^{O(l)}$ and in space $O(n)$; here $l = \log_2 n$. (This time bound

is still substantially lower than the time bound for the straightforward exhaustive search.)

It is easy to see that, for a fixed probability p, the expected run-time of HPA on $G_{n,p}$ is the expected run-time of HPA1 on $G_{n,p}$ plus $o(n)$.

The following claim shows that, in a sense, our algorithm has the best possible expected run-time.

CLAIM. *Let A be any algorithm for the Hamiltonian Path Problem whose input is provided by an oracle: given a pair of vertices u, v as a query, the oracle answers whether $\{u, v\}$ is an edge. For every positive ε there is n_0 such that for every $n \geqq n_0$ and every $p > (3 \log_e n)/n$, the expected number of queries during the run of A on $G_{n,p}$ is at least $(1-\varepsilon)n/p$.*

Proof. Let Q be the number of different queries, and P be the number of different queries answered positively. The expectation $E(P)$ equals $p \times E(Q)$; hence $E(Q) = p^{-1}E(P)$. If there is a Hamiltonian path (from the designated Start to the designated Finish) then $P \geqq n-1$. Since $p > (3 \log_e n)/n$, the probability of the existence of a Hamiltonian path converges to 1 when n grows to infinity (Bollobas (1979), Posa (1976)). □

The algorithm HPA can be adapted to directed graphs. For definiteness, we work with the analogue $\Delta(n, p)$ of $\Gamma(n, p)$ which comprises directed graphs D with vertices $0, 1, \cdots, n-1$ and at most one edge between any two vertices. The probability, assigned to a particular D, is the probability that D is the result of the following experiment: Given distinct vertices u and v, toss an unbiased coin to determine which of the two ordered pairs (u, v), (v, u) is a candidate to be an edge. Having a candidate (x, y), toss a coin with the probability p of the head; declare (x, y) an edge if and only if the head turns up. Let $D_{n,p}$ be a random member of $\Delta(n, p)$. In § 4 we indicate changes necessary to turn HPA to an algorithm HPA_d for solving the Hamiltonian Path Problem for directed graphs.

THEOREM 2. *There is a positive real number c satisfying the following condition. For a fixed probability p, the expected run-time of HPA_d on $D_{n,p}$ is $cn/p + o(n)$.*

The expected run-time of HPA on a $G_{n,p(n)}$ remains polynomial if $p(n)$ changes with n but does not decrease too fast. We do not want to state explicitly the rate of decrease that leaves the expected run-time of HPA on a $G_{n,p(n)}$ polynomial: the algorithm HPA and its exposition were designed with the case of a fixed p in mind. A proper treatment of the case of decreasing $p(n)$ will require a different algorithm.

Conjecture. There is an algorithm for the Hamiltonian Path Problem with polynomial (in n) expected run-time on any $G_{n,p}$.

This paper was published first as a technical report (Gurevich and Shelah (1984)).

0. Preliminaries. Consider two coloring algorithms.

The first coloring algorithm. Given a graph do the following. For every pair $\{u, v\}$ of distinct vertices, toss an unbiased coin (which will be referred to as the red coin). If the coin turns up a head and $\{u, v\}$ is an edge, then declare $\{u, v\}$ a red edge.

The second coloring algorithm. Given a graph do the following. For every pair $\{u, v\}$ of distinct vertices, toss three unbiased coins: an orange coin, a yellow coin, and a green coin. If the orange (resp. yellow, green) coin turns up a head and $\{u, v\}$ is an edge then declare $\{u, v\}$ an orange (resp. yellow, green) edge.

The first coloring algorithm turns the given adjacency matrix M into a "red" adjacency matrix M'. For each pair $\{u, v\}$ of distinct vertices, M' contains one bit of information which tells us whether $\{u, v\}$ is a red edge or not. We introduce a new probability space $\Gamma'(n, p)$. Let G be a graph on $\{0, 1, \cdots, n-1\}$ with adjacency matrix

M and red adjacency matrix M'. The probability of G in $\Gamma'(n, p)$ is the product of the following:

The probability that M is the adjacency matrix of a random member of $\Gamma(n, p)$, and

The probability that, given M, the first coloring produces M'.

The two coloring algorithms together turn the given adjacency matrix M into an extended adjacency matrix M^*. For each pair $\{u, v\}$ of distinct vertices, M^* contains five (dependent) bits of information: the original bit saying whether $\{u, v\}$ is or is not an edge, the "red bit" saying whether $\{u, v\}$ is or is not a red edge, and three other "color bits." We introduce a new probability space $\Gamma^*(n, p)$. Members are graphs with vertices $0, 1, \cdots, n-1$ and both uncolored and colored edges; an edge may have any subset of the colors red, orange, yellow, green. (Formally, members of $\Gamma'(n, p)$ are extended adjacency matrices.) Let G be a graph on $0, 1, \cdots, n-1$ with adjacency matrix M and extended adjacency matrix M^*. The probability of G in $\Gamma^*(n, p)$ is the product of the following:

The probability that M is the adjacency matrix of a random member of $\Gamma(n, p)$, and

The probability that, given M, the two coloring algorithms produce M^*.

Given the adjacency matrix of a graph, we would like to apply the two coloring algorithms first and then to apply the rest of HPA to the resulting extended adjacency matrix. Unfortunately, coloring takes too much time. To get around this problem, HPA tosses a C-coin for a pair of different vertices when the pair is examined for being a C-edge for the first time. Logically, coloring precedes the other parts of the algorithm HPA, but operationally its work will be divided between HPA1 and HPA2: HPA1 will do some of the red coloring, and HPA2 will do all orange, yellow and green coloring.

Remark. One may alter the coloring in different ways. Since HPA1 uses only red edges to construct a Hamiltonian path and since the expected running time of HPA is defined largely by the expected running time of HPA1, one may like to increase the probability of red color. Actually, red coloring can be skipped altogether because only a small fraction of edges is examined by HPA1, but then HPA2 should be substantially modified.

DEFINITION. A *primitive boolean combination* of events $E_1, \cdots, E_m$ is the intersection of events $X_1, \cdots, X_m$ where each X_i is either the event E_i or the complement of the event E_i. An event E is *realizable* if its probability is positive.

The probability that an arbitrary pair $\{i, j\}$ of distinct vertices is a red (respectively, orange, yellow, green) edge is $p/2$.

LEMMA 0.1. *For every $k \in \{1, 2, 3\}$ there is a positive real number c_k (not depending on p or n) which satisfies the following conditions. Suppose that $J = \{$red, orange, yellow, green$\}$, $C \in J$, K is a subset of $J - \{C\}$ of cardinality k, G is a random member of $\Gamma^*(n, p)$, $\{u, v\}$ is a pair of distinct vertices of G, and E is the event "$\{u, v\}$ is an edge of color C." Then*

(a) *The probability of E, assuming any primitive boolean combination of the events "$\{u, v\}$ is an edge of color D," where $D \in K$, is at least $c_k \cdot p$;*

(b) *The probability of E, assuming any realizable boolean combination of the events "$\{u, v\}$ is an edge of color D," where $D \in K$, is at least $c_k \cdot p$;*

(c) *Let H be any realizable boolean combination of the events "$\{x, y\}$ is of color D" excluding E. Then $\mathbf{P}[E|H] \geqq c_k \cdot p$.*

Proof. (a) An easy and direct computation of conditional probabilities establishes the statement.

(b) A realizable boolean combination of some events equals a disjoint sum of primitive boolean combinations of those events. Now use the fact that if Y, Z are disjoint realizable events and $\mathbf{P}[X \mid Y] \geqq r$, $\mathbf{P}[X \mid Z] \geqq r$ then $\mathbf{P}[X \mid Y \cup Z] \geqq r$. We verify

here this simple fact.

$$\begin{aligned}\mathbf{P}[X \mid Y \cup Z] &= \mathbf{P}[X(Y \cup Z)] : \mathbf{P}[Y \cup Z] = (\mathbf{P}[XY] + \mathbf{P}[XZ]) : (\mathbf{P}[Y] + \mathbf{P}[Z]) \\ &= \mathbf{P}[XY] : (\mathbf{P}[Y \cup Z]) + \mathbf{P}[XZ] : (\mathbf{P}[Y \cup Z]) \\ &= \mathbf{P}[X \mid Y] \times \mathbf{P}[Y \mid Y \cup Z] + \mathbf{P}[X \mid Z] \times \mathbf{P}[Z \mid Y \cup Z] \geqq r.\end{aligned}$$

(c) Without loss of generality, H is a primitive boolean combination. Therefore H is the intersection AB where A is an event about $\{u, v\}$ and B is an event about other pairs of vertices. Thus,

$$\mathbf{P}[E \mid AB] = \mathbf{P}[EAB]/\mathbf{P}[AB] = (\text{by virtue of the independence}),$$

$$(\mathbf{P}[EA] \times \mathbf{P}[B])/(\mathbf{P}[A] \times \mathbf{P}[B]) = \mathbf{P}[E \mid A] \geqq c_k \cdot p. \qquad \square$$

The algorithm HPA will use the four colors in the order of spectrum. This motivates the following definition. Let c_1, c_2 and c_3 be the maximal numbers satisfying Lemma 0.1. Then $p_{\text{red}} = p/2$, $p_{\text{orange}} = c_1 \cdot p$, $p_{\text{yellow}} = c_2 \cdot p$, and $p_{\text{green}} = c_3 \cdot p$. For each color C, let $q_C = 1 - p_C$. It is easy to check that $p_{\text{red}} > p_{\text{orange}} > p_{\text{yellow}} > p_{\text{green}}$.

In the rest of this section we prove a couple of auxiliary propositions about probabilities. The first one helps to deal with families of events that are independent only to a certain degree.

PROPOSITION 0.1. *Let* $0 < r < 1$ *and* $0 \leqq \beta \leqq 1$.

(a) *Consider m Bernoulli trials with the probability r of success in a single trial. Then*

$$\mathbf{P}[\textit{at most } (1-\beta)mr \textit{ successes}] \leqq e^{**}(-\beta^2 mr/2),$$

$$\mathbf{P}[\textit{at least } (1+\beta)mp \textit{ successes}] \leqq e^{**}(-\beta^2 mr/3).$$

(b) *Let* $E_1, \cdots, E_m$ *be events in a probability space. Suppose that the probability of each* E_j, *assuming any primitive boolean combination of the events* E_j *with* $j < i$, *is at least r. Then*

$$\mathbf{P}[\textit{at most } (1-\beta)mr \textit{ events } E_i \textit{ happen}] \leqq e^{**}(-\beta^2 mr/2).$$

Proof. (a) We borrow the two inequalities from Angluin and Valiant (1979). They follow from Chernoff's bound (Chernoff (1952)).

(b) Let $D_{j,k,l}$ be the collection of sample points s such that s belongs to at most k among l events $E_{j+1}, \cdots, E_{j+1}$. In virtue of (a), it suffices to prove that for all j, k, l and H, if $1 \leqq l \leqq m$, $0 \leqq j \leqq m - l$, $0 \leqq k \leqq l$ and H is realizable then $\mathbf{P}[D_{j,k,l} \mid H] \leqq \Sigma\{b_{i,l} : 0 \leqq i \leqq k\}$ where $b_{i,l}$ is the probability of i successes in L Bernoulli trials with probability r of success in a single trial.

The proof is by induction on l. Basis of induction:

$$\mathbf{P}[D_{j,0,1} \mid H] = \mathbf{P}[\bar{E}_{j+1} \mid H] \leqq 1 - r = b_{0,1}; \qquad \mathbf{P}[D_{j,1,1} \mid H] \leqq 1 = b_{0,1} + b_{1,1}.$$

Induction step. Let $H_0 = H$, $H_{i+1} = H_i \cap \bar{E}_{j+i+1}$ for $i < l$. If $D_{j,0,l}$ is realizable then all H_i are realizable and $\mathbf{P}[D_{j,0,l} \mid H] = \Pi\{\mathbf{P}[\bar{E}_{j+i+1} \mid H_i : i < 1\} \leqq (1-r)^l = b_{0,l}$; otherwise $\mathbf{P}[D_{j,0,l} \mid H] = 0 \leqq b_{0,l}$. It remains to prove that $\mathbf{P}[D_{j,k+1,l+1} \mid H] \leqq \Sigma\{b_{i,l+1} : 0 \leqq i \leqq k+1\}$. If $\alpha = \mathbf{P}[E_{j+1} \mid H] < 1$ then H_1 is realizable and

$$\begin{aligned}\mathbf{P}[D_{j,k+1,l+1} \mid H] &= \alpha \times \mathbf{P}[D_{j+1,k,l} \mid H \cap E_{j+1}] \\ &\quad + (1-\alpha) \times \mathbf{P}[D_{j+1,k+1,l} \mid H_1] \quad (\text{by the induction hypothesis})\end{aligned}$$

$$\leqq \alpha \times \Sigma\{b_{i,l} : 0 \leqq i \leqq k\} + (1-\alpha) \times \Sigma\{b_{i,l} : 0 \leqq i \leqq k+1\}$$

(which decreases with α and therefore)

$$\leqq r \times \Sigma\{b_{i,l} : 0 \leqq i \leqq k\} + (1-r) \times \Sigma\{b_{i,l} : 0 \leqq i \leqq k+1\}$$

$$= \text{(as in the case when the events } E_i \text{ are truly independent)}$$

$$\Sigma\{b_{i,l+1} : 0 \leqq i \leqq k+1\}.$$

If $\alpha = 1$ skip the second expression in the above computation. □

Note. One of the referees suggested as reference (Graham (1983)) in connection to the proof of Proposition 0.1.

The next proposition is generalization of the Bottleneck Lemma of Angluin and Valiant (1979). It can be used often as an alternative to Proposition 0.1. If S is a set of strings, let $\text{Pref}(S) = \{s' : s' \text{ is a prefix of some string } s \in S\}$. A prefix s' of a string s is *proper* if $|s'| < |s|$.

PROPOSITION 0.2. *Let*

S be a set of strings over some finite alphabet A,

k be a natural number and $0 \leqq p_1, \cdots, p_k \leqq 1$,

E be an arbitrary realizable event,

$\alpha_1, \alpha_2, \alpha_3, \cdots$ be random variables taking values in A.

Suppose that for each $s \in S$ there are proper substrings $s_1, \cdots, s_k$ of s such that $|s_1| < \cdots < |s_k|$, and for each s_i, if $l = |s_i| + 1$ then $\mathbf{P}[s_i \alpha_l \in \text{Pref}(S) \mid E] \leqq p_i$.

Then $\Sigma_{s \in S} \mathbf{P}[\alpha_1 \cdots \alpha_{|s|} = s \mid E] \leqq p_1 \bullet \cdots \bullet p_k$.

Proof. The proof follows by induction on k. The case $k = 0$ is trivial: the product of zero factors is supposed to be equal to 1.

Suppose $k > 0$ and the proposition is proved for $k-1$. For each $s \in S$ fix some appropriate substrings $s_1, \cdots, s_k$; let $T = \{s_k : s \in S\}$. If $t \in T$ then there are proper substrings $t_1, \cdots, t_{k-1}$ of t such that $|t_i| < \cdots < |t_{k-1}|$, and for each t_i, if $l = |t_i| + 1$ then $\mathbf{P}[t_i \alpha_l \in \text{Pref}(T) \mid E] \leqq p_i$. If $t \in T$ then $\mathbf{P}[t\alpha_{|t|+1} \in \text{Pref}(S) \mid E] \leqq p_k$. Thus,

$$\Sigma_{s \in S} \mathbf{P}[\alpha_1 \cdots \alpha_{|s|} = s \mid E] \leqq \Sigma_{t \in T} (\mathbf{P}[\alpha_1 \cdots \alpha_{|t|} = t \mid E] \times \mathbf{P}[t\alpha_{|t|+1} \in \text{Pref}(S) \mid E])$$

$$\leqq (\Sigma_{t \in T} \mathbf{P}[\alpha_1 \cdots \alpha_{|t|} = t \mid E]) \times p_k \leqq p_1 \bullet \cdots \bullet p_{k-1} \cdot p_k. \quad \square$$

Following Angluin and Valiant, we explain the intuition behind Proposition 0.2. Ignore event E and suppose that $p_1 = \cdots = p_k = p$. Think about $\text{Pref}(S)$ as the set of, say, violet nodes on the tree of strings over A. Say that a violet node t is a *bottleneck* if the probability of drawing a violet successor of t is at most p. The proposition says that if there are at least k bottlenecks on the way to each S-node then the probability to arrive from the root to S is at most p^k.

1. Algorithm HPA1. The algorithm HPA1, described in this section, attempts to construct an (all-red) Hamiltonian path in a given graph from one distinguished vertex Start to another distinguished vertex Finish. HPA1 does not attempt to establish the nonexistence of a Hamiltonian path.

To simplify the exposition, we assume that the given graph is colored red and the red adjacency matrix is given. The assumption will be removed at the end of the section.

The description of HPA1 is interleaved with some analysis. We fix a probability p and analyse the run of HPA1 on a random member of some $\Gamma'(n, p)$. We will prove that the expected run-time is $cn/p + o(n)$ for some absolute (i.e. not depending on p) constant c, and that, almost surely, HPA1 succeeds in constructing an all-red Hamiltonian path. In this section, an event $E(n)$ will be called *negligible* if its probability is $2^{**}(-O/(\sqrt{n}))$, and *almost sure* if its complement $\bar{E}(n)$ is negligible.

Remark. The bound $2^{**}(-O(\sqrt{n}))$ is chosen somewhat arbitrarily. One can play with this bound. For example, $\sqrt{n}$ can be replaced by $(\log_2 n)^2$ or any n^ε with $\varepsilon < 1$; HPA1 is easily adjustable. However, the algorithm HPA1 definitely fails if G does not have a Hamiltonian path from Start to Finish. Thus the probability of failure is not less than the probability $(1-p)^{n-1}$ of a vertex to be isolated.

The algorithm HPA1 works in several stages.

Stage 0. Create four lists:
Even, consisting of only one vertex, namely the vertex Start;
Odd, consisting of only one vertex, namely the vertex Finish;
Even Outback, consisting of all remaining even vertices; and
Odd Outback, consisting of all remaining odd vertices.

To simplify exposition, we ignore rounding reals (i.e. we ignore necessary applications of the floor and ceiling functions $\lfloor \cdots \rfloor$ and $\lceil \cdots \rceil$).

Stage 1. Extend Even by means of sweeps through Even Outback. During one sweep, for each Even Outback vertex v in turn, try to extend Even by means of v: if the pair {Last (Even), v} is a red edge then remove v from Even Outback and make it the new Last (Even). Halt when an idle sweep (a sweep that does not extend Even) is discovered or Even Outback is emptied. If Even Outback contains $\geqq\sqrt{n}$ vertices then go to HPA2.

LEMMA 1.1. *The expected time of Stage* 1 *is* $cn/p + o(n)$ *for some* c, *and almost surely* HPA1 *succeeds on Stage* 1, *i.e. almost surely* $<\sqrt{n}$ *vertices are left in Even Outback.*

Proof. On stage 1, HPA1 repeatedly makes attempts to extend Even by an Even Outback vertex. The total number of attempts is bounded by n^2. The expected number of attempts is $n/p_{\text{red}} + o(n)$. For, if an idle sweep is not discovered within n/p_{red} attempts, then almost surely Even acquires all even vertices: use Proposition 0.1 with $m = n/p_{\text{red}}$, $\beta = \frac{1}{2}$ and $r = p_{\text{red}}$. Since every attempt takes only bounded many steps, the expected time of Stage 1 is $cn/p + o(n)$ for some c.

Suppose that $\geqq\sqrt{n}$ vertices are left in Even Outback after Stage 1. It means that there is a vertex x (the final Last (Even)) such that examining $\geqq\sqrt{n}$ different pairs $\{x, v\}$, not examined earlier, reveals that none of them is a red edge. The probability of such an event is $\leqq n \times (q_{\text{red}}{}^{**}\sqrt{n})$ so that the event is negligible. □

Stage 1′. Extend Odd similarly, but place Odd Outback vertices at the beginning of Odd rather than at the end. In particular, Finish remains the last member of Odd.

Stage 2. Concatenate an initial segment of the current list Even with a final segment of the current list Odd into one path P. Let x_i be the ith element of Even from the end, and let y_j be the jth element of Odd (from the beginning). Try pairs $\{x_0, y_0\}$, $\{x_0, y_1\}$, $\{x_1, y_0\}$, etc. one after another to find one that is a red edge. The order is first by the sum of indices and then lexicographically. Halt when a red edge $\{x_i, y_j\}$ comes along or when $i+j$ reaches $\sqrt{n}$. If a red edge $\{x_i, y_j\}$ is discovered, then use the edge to concatenate the initial segment [Start, x_i] of Even and the final segment [y_j, Finish] of Odd into one Path P, then return $x_1, \cdots, x_{i-1}$ to Even Outback, and return $y_1, \cdots, y_{j-1}$ to Odd Outback. If $i+j$ reaches $\sqrt{n}$ go to HPA2.

LEMMA 1.2. *Almost surely,* HPA1 *succeeds on Stage* 2 (*if it arrives there*). *The expected time of Stage* 2 *is bounded by a constant* (*which depends on* p).

Proof. The proof is clear. □

Stage 3. Insert Even Outback vertices to P, one by one, by means of one sweep through P. In the following procedure, Pred is the predecessor function defined by P.

1. Set $x :=$ Last (P);
2. If Even Outback is empty then halt else set $v :=$ First (Even Outback);

3. If x is one of the first 4 vertices in P then go to HPA2;
4. If both $\{\text{Pred}(x), v\}$ and $\{v, x\}$ are red edges
   then place v into P between Pred (x) and x, set $x := \text{Pred}(x)$,
   drop v from Even Outback, go to 2
   else if $\{\text{Pred}(x), v\}$ is a red edge then set $x := \text{Pred}(x)$, go to 3
   else set $x := \text{Pred}(\text{Pred}(x))$, go to 3.

LEMMA 1.3. *The run-time of Stage* 3 *is bounded by some cn where c does not depend on p. Almost surely,* HPA1 *succeeds on Stage* 3 *(if it arrives there).*

Proof. The first statement is obvious. We prove the second.

First let us review the situation at the beginning of Stage 3. The Even Outback contains less than $2\sqrt{n}$ vertices which are all even. P has a tail of at odd vertices plus Finish, the length of the tail is at least $(n/2)-2\sqrt{n}$. If v is an Even Outback vertex and x belongs to the tail of P then the pair $\{v, x\}$ was not looked at earlier, hence $\mathbf{P}[\{v, x\} \text{ is a red edge}] = p_{\text{red}}$. Let $m = (1/2) \times ((n/2)-2\sqrt{n})$.

On Stage 3, HPA1 repeatedly makes attempts to insert an Even Outback vertex into P. Let E_i be the event that either HPA1 does not make the ith attempt (i.e. Even Outback was emptied during previous attempts) or HPA1 makes the ith attempt and the ith attempt is successful. Obviously, the probability of E_i, assuming any primitive boolean combination of the previous events, is at least $r = p_{\text{red}}^2$. By Proposition 0.1, almost surely more than $mr/2$ events E_i happen. Hence, almost surely HPA1 succeeds on Stage 3. □

Stage 3′. Similarly insert the Odd Outback vertices in P.

Let us summarize the analysis of HPA1 in the following theorem.

THEOREM 1.1. *There is a positive real number c satisfying the following condition: Fix a probability p. Let G be a random member of the probability space* $\Gamma'(n, p)$ *for some n. The expected run-time of* HPA1 *on G is* $cn/p + o(n)$, *and the probability that* HPA1 *fails on G is* $2^{**}(-O(\sqrt{n}))$.

Proof. The proof is clear. □

To simplify the exposition, we have supposed above that the input to HPA1 is a colored graph whereas in effect the input is an ordinary graph and HPA1 should perform the red coloring.

THEOREM 1.2. *There is a positive real number c satisfying the following condition. Fix a probability p. Let G be a random member of the probability space* $\Gamma(n, p)$ *for some n. The expected run-time of* HPA1 *on G is* $cn/p + o(n)$, *and the probability that* HPA1 *fails on G is* $2^{**}(-O(\sqrt{n}))$.

Proof. It is easy to see that our analysis of HPA1 remains valid; in particular the three lemmas remain valid. The only difference is that the minimal appropriate constant c should be a bit larger to account for time spent for red coloring. □

2. Algorithm HPA2. The polynomial time algorithm HPA2, described in this section, attempts to construct or to establish the nonexistence of a Hamiltonian path in a given graph from one distinguished vertex Start to another distinguished vertex Finish.

To simplify the exposition, we suppose that the given graph is colored and is given by means of the extended adjacency matrix. The assumption will be removed at the end of this section. Our estimations of the run-time of HPA2 and the chances of HPA2 to succeed will remain true under the assumption of any realizable boolean combination of the events "$\{u, v\}$ is a red edge." In particular, they will remain true under the assumption that HPA1 fails on G, which is a boolean combination of the events "$\{u, v\}$ is a red edge" because HPA1 works only with red edges.

Again, the description of the algorithm is interleaved with some analysis. We fix a probability p and analyse the run of HPA2 on a random member G of the probability space $\Gamma^*(n, p)$ where n is arbitrary but not too small; we will take for granted, for example, that $n > 2 + 3/(-\log_2 q_{\text{green}})$.

HPA2 uses orange, yellow and green colors to construct a Hamiltonian path, and works in several stages. As in § 1, we will often ignore rounding reals in order to simplify exposition. Let V be the set of vertices of G and $l = \log_2 n$.

Stage 1. Compute the set T (for "troublesome") of vertices with small orange, yellow or green fan. For each vertex v, let $M(v) = \min$ {the number of C-edges incident to v: C is orange, yellow or green}. Compute $T' = \{v: M(v) < nl^{-2}\}$. If $|T'| \geqq 3/(-\log_2 q_{\text{green}})$ then go to HPA3, else set $T := T' \cup$ {Start, Finish} and output T.

LEMMA 2.1. *The time of Stage* 1 *is* $O(n^2)$. *The probability that* HPA2 *fails on Stage* 1 (*i.e.* $|T'| \geqq 3/(-\log_2 q_{\text{green}})$) *is bounded by* $2^{-3n+o(n)}$.

Proof. The first statement is obvious; we prove the second.

Let E be any event "$\{u, v\}$ is a C-edge" where u, v are distinct vertices and $C \in$ {orange, yellow, green}, and let $q = q_{\text{green}}$. By Lemma 0.1 and the definition of q_{green}, the probability of $\bar{E}$ (which is the complement of E), assuming any realizable boolean combination of the events "$\{x, y\}$ is a D-edge" different from E, is at most q.

Let $m = nl^{-2}$. For each vertex $v \in T'$, there are a color $C \in$ {orange, yellow, green} and a set $S \subseteq V - \{v\}$ of at most $m - 1$ vertices such that there are no C'-edges between v and $(V - \{v\}) - S$. Hence,

$$\mathbf{P}[v \in T'] \leqq 3\binom{n-1}{m-1} q^{n-m} \leqq 3n^{m-1} q^{n-m},$$

$$\mathbf{P}[|T'| \geqq k] \leqq \binom{n}{k} (3n^{m-1} q^{n-m})^k \leqq n^k (3n^{m-1} q^{n-m})^k \leqq (3n^m q^{n-m})^k,$$

and

$$\log_2 \mathbf{P}[|T'| \geqq k] \leqq k[\log_2 3 + ml - (n-m)(-\log_2 q)].$$

Hence $\log_2 \mathbf{P}[|T'| \geqq 3/(-\log_2 q)] \leqq -3n + o(n)$. □

Stage 1 of HPA2 is a special procedure which outputs the set T of troublesome vertices. The extended adjacency matrix and the set T constitute the input for the rest of HPA2. Let $\Gamma^*_T(n, p)$ be the subspace of $\Gamma^*(n, p)$ defined by the given set T; a member A of $\Gamma^*(n, p)$ belongs to $\Gamma^*_T(n, p)$ if and only if the given T is the output of Stage 1 applied to A. In the rest of this section, we work with a random member G of $\Gamma^*_T(n, p)$.

PROPOSITION 2.1. *Let numbers* c_1, c_2 *and* c_3 *satisfy Lemma* 0.1. *Suppose that*

$J =$ {red, orange, yellow, green} *and* $C \in$ {orange, yellow, green},

K *is a subset of* $J - \{C\}$ *and* $k = |K|$,

$\{u, v\}$ *is a pair of distinct vertices in* $V - T$,

E *is the event "*$\{u, v\}$ *is an edge of color* C,"

H *is a realizable boolean combination of events "*$\{x, y\}$ *is of color* D" *excluding* E.

Then $\mathbf{P}[E \mid H] \geqq c_k \cdot p$.

Proof. Without loss of generality, we may assume that H is a primitive boolean combination: see the proof of statement (b) of Lemma 0.1. Let a be the cardinality of the set $\{x \in V - \{u, v\}: \{u, x\}$ is an edge of color C with respect to $H\}$, and b be the cardinality of the set $\{x \in V - \{u, v\}: \{v, x\}$ is an edge of color C with respect to $H\}$. Let $m = nl^{-2}$. If $a < m - 1$ then $u \in T$ which is impossible; hence $a \geqq m - 1$. Similarly, $b \geqq m - 1$. If $a = m - 1$ or $b = m - 1$ then $\mathbf{P}[E \mid H] = 1$. Suppose $a \geqq m$ and $b \geqq m$. Without loss of generality, in the probability space $\Gamma^*(n, p)$, hypothesis H implies the

event $G \in \Gamma_T^*(n, p)$. Therefore the desired conditional probability $\mathbf{P}[E \mid H]$ can be computed in $\Gamma^*(n, p)$. Now use statement (c) of Lemma 1 in § 0. □

In order to motivate the next definition, let us notice that any Hamiltonian path P and any set U of vertices give rise to a family $\{P_u: u \in U\}$, where P_u is the maximal segment of P such that $u \in P$ and U contains at least one of any two neighboring elements of P.

DEFINITION. Let U be a set of vertices. A family F of disjoint (having no common vertices) paths is an *envelope* for U if

(a) $U \subseteq \cup F$;

(b) Every U-vertex, different from Start and Finish, is an internal vertex of some F-path;

(c) If Start belongs to U then it is the first element of some F-path, if Finish belongs to U then it is the last element of some F-path, and if both Start and Finish belong to U then they belong to different F-paths unless $U = V$ and $|F| = 1$;

(*d*) If u, v are neighbors in some F-path then $u \in U$ or $v \in U$.

Stage 2. Construct an envelope F for T, or establish the nonexistence of such an envelope. The desired envelope is constructed by brute force but we should be a little careful: even though T is small, there may be relatively many vertices adjacent to T-vertices. For each vertex v, let Fan (v) be the number of all edges incident to v.

1. Set $X := T$ and List := Empty.
2. While there is $v \in X$ with Fan $(v) \geqq 3|X|$ do the following:
   Make one such v the head of List and set $X := X - \{v\}$.
3. Find an envelope F for X or the nonexistence of such an envelope by exhaustive search. (This is possible because there are $\leqq 3|T|^2$ vertices v such that v belongs to X or is adjacent to a vertex in X.) In the negative case write "There is no Hamiltonian path from Start to Finish," and halt.
4. While List $\neq$ Empty do the following:
   4.1. Set $v :=$ Head (List), List := Tail (List).
   4.2. Extend F to an envelope for $X \cup \{v\}$. Notice that $\cup F$ contains at most $3|X|$ vertices whereas v is adjacent to at least $3(|X|+1)$ vertices. If $v =$ Start and v is the first vertex of some $P \in F$, or $v =$ Finish and v is the last vertex of some $P \in F$, or v is an internal vertex of some $P \in F$ then do nothing, otherwise do the following. If v is an end of some P in F then pick $u \in V - \cup F$ adjacent to v, and extend P by the edge $\{v, u\}$. If $v \notin \cup F$ then pick
   $u \in \{x: x \in V - \cup F, x \text{ is adjacent to } v, \text{ and } x \neq \text{Finish}\}$,
   $w \in \{x: x \in V - \cup F, x \text{ is adjacent to } v, x \neq \text{Start, and } x \neq u\}$
   and extend F by a new path (u, v, w).
   4.3. Set $X := X \cup \{v\}$.

LEMMA 2.2. *The time of Stage* 2 *is* $O(n)$.

Proof. The proof is clear. □

Remark. In connection to the case of decreasing $p(n)$, one of the referees was interested in the run-time of step 3. We describe a variation of the dynamic programming algorithm (Bellman (1960); Held and Karp (1962)) to carry out step 3. Let $U = \{v \in V - T: v \text{ is adjacent to a vertex in } T\}$. A chain P of vertices from $T \cup U$ will be called a *pseudo-path* if (a) Start is the first vertex in P, (b) if v is the successor of u in P but $\{u, v\}$ is not an edge then $\{u, v\} \subseteq U$, there is an edge between u and its P-predecessor, and there is an edge between v and its P-successor unless v is the last in P, (c) if v is the successor of u in P and $\{u, v\}$ is an edge then $\{u, v\} \cap T \neq \emptyset$, (d) if Finish belongs to P then $T \subseteq P$ and Finish is the last in P. The idea is to compute

a function f whose properties are described below. The domain of f consists of triples (X, Y, v) where $\{\text{Start}\} \subseteq X \subseteq T$, $Y \subseteq U$, $|Y| \leqq 2|X|$, and $v \in X \cup Y$. Each $f(X, Y, v)$ is either a pseudo-path P such that $X \subseteq P$ and v is the last in P, or the string "The desired pseudo-path does not exist." Compute f by induction on $|X|$. Turn any pseudo-path $f(T, Y, \text{Finish})$ into the desired envelope for T; if no $f(T, Y, \text{Finish})$ is a pseudo-path then write "There is no Hamiltonian path from Start to Finish" and halt. The run-time of this algorithm is bounded by |domain (f)| times a polynomial in n.

Stage 3. Extend F-paths using orange edges. Let $P_1, \cdots, P_k$ be the F-paths where Start = First (P_1), and Finish = Last (P_k). Obviously, $k \leqq |T|$. The orange fans of vertices in $V - T$ are at least n/l^2; in particular, the orange fans of Last $(P_1), \cdots,$ Last (P_{k-1}), First $(P_2), \cdots,$ First (P_k) are at least n/l^2. The union of the paths $P_1, \cdots, P_k$ contains less than $3|T|$ vertices. Find the maximal m such that $3|T| + 2m(k-1) \leqq n/l^2$. Using only orange edges, extend each path $P_2, \cdots, P_k$ by an additional final segment of length m and extend each path $P_1, \cdots, P_{k-1}$ by an additional initial segment of length m in such a way that the extended paths $P_1, \cdots, P_k$ remain disjoint.

LEMMA 2.3. *The time of Stage* 3 *is* $O(n)$.

Proof. The proof is obvious. □

Stage 4. Sew the paths $P_1, \cdots, P_k$ into one path Q_0 using yellow edges. For $i < k$, let A_i be the final tenth part of P_i; for $j > 1$, let B_j be the initial tenth part of P_j. For each $i < k$, find $x_i \in A_i$ and $y_i \in B_{i+1}$ such that $\{x_i, y_i\}$ is a yellow edge, then throw from P_i the successors of x_i and throw from P_{i+1} the predecessors of y_i. In the case of failure (for at least one i) go to the algorithm HPA3; in the case of success sew the (possibly shortened) paths $P_1, \cdots, P_k$ into one path Q_0 using the yellow edges $\{x_i, y_i\}$.

LEMMA 2.4. *The time of Stage* 4 *is* $O(n^2/l^{-4})$. *The probability of failure is bounded by* $2^{**}O(-n^2/l^{-4})$.

Proof. The proof is clear. The expected computation time is bounded by a constant (which depends on p). □

Stage 5. Partition $V - Q_0$ into disjoint paths $Q_1, Q_2, \cdots$ satisfying the following condition: either Q_{i+1} is a circuit (a closed path) of length at least $nl^{-2}/2$ or each of the two end-points of Q_{i+1} is adjacent to at least $nl^{-2}/2$ vertices in $\bigcup_{j \leqq i} Q_j$. Assuming that $Q_1, \cdots, Q_i$ have been constructed, $U_i = \bigcup_{j \leqq i} Q_j$, and $V - U_i \neq \varnothing$, we construct Q_{i+1} using, say, yellow edges.

Pick a vertex in $V - U_i$ and set X equal to the path consisting of that vertex only. While there is a yellow edge from Last (X) to a vertex in $V - (U_i \cup X)$, append X by such a vertex. While there is a vertex in $V - (U_i \cup X)$ with a yellow edge to First (X), pick one such vertex and make it the new first vertex of X.

If there are at least $nl^{-2}/2$ yellow edges between First (X) and U_i as well as between Last (X) and U_i, then set $Q_{i+1} := X$. Otherwise let u be an end-point of X with less than $nl^{-2}/2$ yellow edges between u and U_i. Without loss of generality, $u =$ First (X). There are more than $nl^{-2}/2$ vertices v of X such that $\{v, u\}$ is a yellow edge; let w be the last (with respect to X) among these vertices. Turn X into a circuit Q_{i+1} by throwing away the successors of w and adding the edge $\{w, u\}$.

LEMMA 2.5. *The time of Stage* 5 *is* $O(n)$.

Proof. The proof is clear. □

Before we explain the last stage of HPA2, let us introduce some terminology and notation. If P is an open path P, $v \in P$ and $v \neq$ Last (P) then v' is the successor of v with respect to P. We suppose that every closed path (circuit) comes with a distinguished member, called the *initial member*, and a specified direction of traversing the path. Since our vertices are natural numbers, we can use, for example, the following rule:

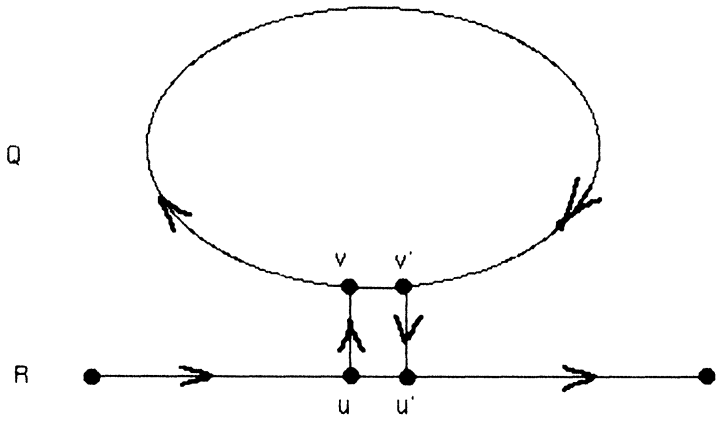

FIG. 1

the member of a circuit with the least numerical value is the initial member, and the lesser of the two neighbors of the initial member defines the direction of traversing. If P is a circuit and $v \in P$ then v' is the predecessor of v with respect to P.

After Stage 5 we have a sequence $Q_0, Q_1, \cdots, Q_m$ of paths. Let Q_{m+1} be the empty set. If $Q = Q_i$ and $i \leqq m$ let $Q' = Q_{i+1}$.

Stage 6. Merging the paths $Q_0, Q_1, \cdots, Q_m$ into the desired Hamiltonian path.

1. Set $R := Q_0$ and $Q := Q_1$.
2. If Q is empty then halt.
3. If Q is a circuit then do the following:
   - 3.1. Set u equal to the first (with respect to R) member of $\{x \in R - T: x' \notin T\}$, and v equal to the initial member of Q.
   - 3.2. If $\{u, v\}$ and $\{v', u'\}$ are green edges, then set R equal to the concatenation of:
     the initial segment [Start, u] of R,
     the segment $[v, v']$ of Q (which includes all vertices of Q), and
     the final segment $[u'$, Finish] of R,
     set $Q := Q'$, and go to 2. (See Fig. 1.)
   - 3.3. Find or establish the nonexistence of a pair (x, y) such that $x \in R - T$, $x' \notin T$, $y \in Q$, and neither of the pairs $\{x, y\}$, $\{y', x'\}$ has yet been examined for being a green edge. In the positive case set $u := x$, $v := y$ and go to 3.2; in the negative case go to HPA3.
4. If Q is an open path, then do the following.
   - 4.1. Set
     $A := \{w \in R - T: w' \notin T$, and $\{w,$ First $(Q)\}$ is a yellow edge$\}$,
     $B := \{w \in R - T: w' \notin T$, and $\{$Last$(Q), w\}$ is a yellow edge$\}$,
     I equal to the least initial segment of R that contains one half of A or one half of B.
     If I does not contain one half of A, then set Q equal to the reverse of Q, and swap A and B.
     (When step 4.1 is completed, $A = \{w \in R - T: w' \notin T$, and $\{w,$ First $(Q)\}$ is a yellow edge$\}$, I is an initial segment of R which contains at least one half of A, $B = \{w \in R - T: w' \notin T$, and $\{w,$ Last $(Q)\}$ is a yellow edge$\}$, and $R - I$ contains at least one half of B.)

4.2. Set
u equal to the first (with respect to R) member of $A \cap I$,
v equal to the first (with respect to R) member of $B - I$.
4.3. If $\{u', v'\}$ is a green edge then
set R equal to the concatenation of:
the segment [Start, u] of R,
the path Q,
the reverse of the segment $[u', v]$ of R,
the segment $[v', \text{Finish}]$ of R,
set $Q := Q'$, and go to 2. (See Fig. 2.)
4.4. Find or establish the nonexistence of a pair (x, y) such that $x \in A$, $y \in B$, and the pair $\{x', y'\}$ was not examined on being a green edge yet. In the positive case set $u := x$, $v := y$ and go to 4.3; in the negative case go to HPA3.

LEMMA 2.6. *The computation time of Stage* 6 *is* $O(n^2)$. *The probability of the failure on Stage* 6 *is bounded by* $2^{**}O(-n^2/l^{-4})$.

Proof. On Stage 6, HPA2 repeatedly makes attempts to combine the current paths Q and R into a new path R. Each attempt is uniquely identified by a pair (u, v) of vertices. Therefore the number of attempts is bounded by n^2. This gives the bound on the computation time.

It remains to prove the bound on the probability of failure. There is a fraction m of n/l^2 such that in no case HPA2 transfers the control to HPA3 within first m attempts. For $i = 1, \cdots, m$ let E_i be the event that either HPA2 does not make the ith attempt (i.e. the desired Hamiltonian path is constructed during previous attempts) or HPA2 makes the ith attempt and the ith attempt is successful. The probability of E_i, assuming any primitive boolean combination of events E_j with $j < i$, is at least $r = p^2_{\text{green}}$. By Proposition 0.1, the probability that at most $mr/2$ events E_i happen, is $2^{**}O(-n^2/l^{-4})$. Hence the probability that HPA2 fails is $2^{**}O(-n^2/l^{-4})$. □

Note. Reading the technical report (Gurevich and Shelah (1984)), one of the referees expressed the following concern. Suppose that we are merging R with $Q = Q_i$ which happened to be an open path. The set $A \cap I$ contains $\Omega(n/l^2)$ elements, and the set $B \cap (R-1)$ contains $\Omega(n/l^2)$ elements. Thus we have $\Omega(n^2/l^4)$ pairs to search on step 4.4. However, many of those pairs could be examined earlier, when we tried to merge R with those Q_j that $j < i$ and Q_j is an open path. Are there enough pairs left to guarantee the high chances of success in merging R and Q_i?

It is true that Stage 6 splits into the substages of merging R and Q_1, R and Q_2, etc. But consider Stage 6 as one sequence of attempts to merge the current R and the current Q. If HPA2 fails on Step 6, then the great majority of the attempts were

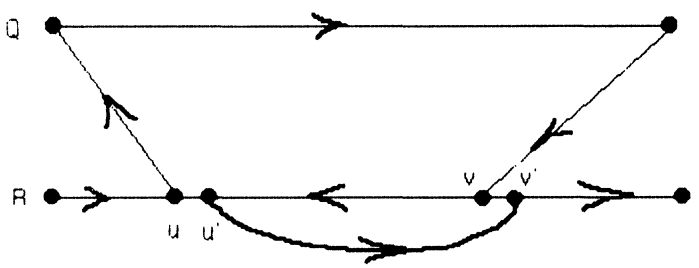

FIG. 2

unsuccessful. We do not care how the unsuccessful attempts were spread over different substages. By Proposition 0.1, such an event is unlikely.

Let us summarize the analysis.

THEOREM 2.1. *The run-time of* HPA2 *on a graph with n vertices is* $O(n^2)$.

Proof. Use Lemmas 2.1–2.6. □

THEOREM 2.2. *Fix a probability p. The probability that* HPA2 *fails on a random member G of a probability space* $\Gamma^*(n, p)$ *is* $2^{-3n+o(n)}$.

Proof. Use Lemmas 2.1, 2.4 and 2.6. □

To simplify the exposition, we have assumed that the input is a random member of $\Gamma^*(n, p)$ whereas in effect we are interested in an input which is the output of HPA1. Instead of a fully colored graph, given by the extended adjacency matrix, we have a graph which is only partially colored red. In addition, we have important information that HPA1 has failed on the corresponding uncolored graph. Let HPA1(G) be the outcome of HPA1 on the input G in the case that HPA1 fails of G.

THEOREM 2.3. *Fix a probability p. Let G be a random member of some* $\Gamma(n, p)$. *The probability that* HPA2 *fails on* HPA1(G), *assuming that* HPA1 *has failed on G, is* $2^{-3n+o(n)}$.

Proof. The idea is that the analysis above remains valid. The coloring is completed on Stage 1 of HPA2. This does not change our estimation of the computation time of Stage 1, and it does not matter for the rest of HPA2 when the graph was colored.

Further, HPA2 never looks to the red part of the extended adjacency matrix. It is easy to see that all six lemmas remain valid under the assumption of any realizable boolean combination of events "$\{u, v\}$ is a red edge." In particular, they remain valid under the assumption that HPA1 fails on the original input. This particular assumption is a boolean combination of events "$\{u, v\}$ is a red edge" because HPA1 utilizes only red edges. □

3. Algorithm HPA3. One obvious algorithm for the Hamiltonian Path Problem is a straightforward exhaustive search: examine in turn all permutations of the vertices different from Start and Finish. Stirling's formula gives an upper bound 2^{nl} on the run-time of the straightforward exhaustive search algorithm on graphs with n vertices; in this section $l = \log_2 n$.

The fastest known algorithm for the Hamiltonian Path Problem is a version of the so-called dynamic programming algorithm (Bellman (1960), Held and Karp (1962)). We sketch the idea roughly. Suppose for simplicity that Start $\neq$ Finish. We will compute a function f satisfying the following conditions. The domain of f consists of pairs (X, v) where X is a set of vertices, Start belongs to X, Finish does not belong to X unless X contains all vertices, v is an element of X different from Start, and if X contains all vertices then $v =$ Finish. Each $f(X, v)$ is either a Hamiltonian path in X from Start to v or the string "The desired Hamiltonian path does not exist." The computation proceeds by induction in $|X|$. The run-time of this algorithm on graphs with n vertices is bounded by 2^n times a polynomial in n. Unfortunately, this algorithm requires exponential space (and a computer with addresses of length about n).

Combining the idea of the dynamic programming algorithm with the idea of Savitch's theorem (Savitch (1970)), we give here a recursive algorithm HPA3 for the Hamiltonian Path Problem. The run-time of HPA3 on a graph with n vertices is bounded by 2^{2n} times a polynomial in n; it is slower than the dynamic programming algorithm but still much faster than the exhaustive search algorithm. The work space of HPA3 on graphs with n vertices is $O(n)$.

We will describe HPA3 informally. Let $(G, \text{Start}, \text{Finish})$ be an instance of Hamiltonian Path Problem, and V be the set of vertices of the graph G. Without loss of generality, $V = \{0, 1, \cdots, n-1\}$ for some n. As in the previous sections, we will often ignore rounding reals in order to simplify exposition.

If $n \leqq 3$ then use the exhaustive search algorithm. Suppose that $n > 3$.

Let Center range over the set $V - \{\text{Start}, \text{Finish}\}$,

$U = V - \{\text{Start}, \text{Center}, \text{Finish}\}$,

A range over subsets of U such that $|A| = |U|/2$ and $\min(U) \in A$ unless $U = \varnothing$,

$B = U - A$,

A' be the subgraph of G with the universe $A \cup \{\text{Start}, \text{Center}\}$,

B' be the subgraph of G with the universe $B \cup \{\text{Center}, \text{Finish}\}$.

For each pair (Center, A) in turn recursively apply HPA3 to the triples $(A', \text{Start}, \text{Center})$ and $(B', \text{Center}, \text{Finish})$. If HPA3 constructs a Hamiltonian path in A' from Start to Center and a Hamiltonian path in B' from Center to Finish, then concatenate the two paths into a Hamiltonian path in G from Start to Finish and halt. Otherwise go to the next pair (Center, A). If there is no next pair, then write "There is no Hamiltonian path from Start to Finish" and halt.

Obviously, the algorithm HPA3 solves the Hamiltonian Path Problem.

THEOREM 3.1. *Let $S(n)$ be the work space and $T(n)$ be the time that* HPA3 *needs in order to handle graphs with n vertices.*

(a) *In an appropriate implementation of* HPA3, $S(n) = O(n)$.

(b) $T(n) = 2^{2n} \times n^{O(1)}$.

Proof. (a) The idea is to pass A (resp. B) to the recursive procedure in the form of the characteristic function which is a boolean vector. The recursive procedure passes further a shorter boolean vector which is a characteristic function that refers to the natural order of A (resp. B). There is a constant b such that for all $n > 3$,

$$S(n) \leqq bn + S((n-3)/2+2) \leqq bn + S(n/2+1).$$

Choose a constant c such that $S(n) \leqq cn$ for $n \leqq 3$, and $bn + c(n/2+1) \leqq cn$ for $n > 3$.

Check by induction on n that $S(n) \leqq cn$ for all n.

(b) There are $n-2$ possible values of Center. For each Center, the number of possible values for A or B is bounded by 2^{n-3}. For each pair (Center, A), HPA3 calls the recursive procedure twice. There is a constant b such that for all $n > 3$,

$$T(n) \leqq 2^{n+bl} \times T(n/2+1).$$

Choose a constant c such that $T(n) \leqq 2^{2n+cl}$ for $n \leqq 3$, and $2^{n+bl} \times 2^{(2n+cl)/2+1} \leqq 2^{n+cl}$ for $n > 3$.

Check by induction on n that $T(n) \leqq 2^{n+cl}$ for all n. □

Remark. Why is HPA3 faster than the straightforward exhaustive search algorithm? To answer this question we use the notation of HPA3. For given Center and A, we spend some time on A *plus* some time on B whereas the straightforward exhaustive search goes through all permutations of A *times* all permutations of B.

Theorems 1.2, 2.2 and 3.1 imply Theorem 1 of the Introduction.

Remark. It is important that the product of the probability of HPA2 to fail and the time bound of HPA3 is $o(n)$. This explains the expression $3/(-\log_2 q_{\text{green}})$ on Stage 1 of HPA2; it guarantees the probability $2^{-3n+o(n)}$ of HPA2 to fail.

4. The case of directed graphs. In this section we prove Theorem 2 of the Introduction. The adaptation of HPA to directed graphs is pretty obvious almost everywhere. The exceptions are Stage 2 and step 4 of Stage 6 of HPA2 which are treated below.

Before we turn to the two exceptions let us notice that Stage 1 of HPA2 should be modified to ensure that T contains all vertices v such that either the orange, yellow or green fan-in of v is less than n/l^2 or the orange, yellow or green fan-out of v is less than n/l^2.

First, we modify Stage 2 of HPA2. Notice that if $v \in X$, and Fan-in $(v) \geqq 3|X|$ unless $v =$ Start, and Fan-out $(v) \geqq 3|X|$ unless $v =$ Finish, then every envelope for $X - \{v\}$ can be easily extended to an envelope for X. Hence the problem reduces to finding or establishing the nonexistence of an envelope for some subset S of T such that for every $v \in S$, either Fan-in $(v) < 3|S|$ or Fan-out $(v) < 3|S|$. Let Y range over minimal sets of edges such that (a) if $v \in S$, $v \neq$ Start and Fan-in $(v) < 3|S|$ then Y contains an edge $\{u, v\}$, and (b) if $v \in S$, $v \neq$ Finish and Fan-out $(v) < 3|S|$ then Y contains an edge $\{v, w\}$. There are only bounded many of different Y's. The problem reduces to the following: given a set Y, find or establish the nonexistence of an envelope F for S such that every Y-edge is a part of an F-path. The Y-edges form a family of paths. If one of the paths is a circle, then the task is impossible, otherwise the task is easy.

Second, we modify step 4 of Stage 6. This time we cannot reverse the order of Q or any segment of R. Set A, B and I as in 4.1. We are forced to consider two cases.

Case 1. I contains one half of A (and therefore $R - I$ contains at least one half of B). Find or establish the nonexistence of a triple (u, v, w) such that $u \in A \cap I$, v is a nonfinal vertex of Q, $w \in B \cap (R - I)$, and (v, u'), (w, v') are edges. In the negative case go to HPA3. In the positive case set R equal to the concatenation of segment [Start, u] of R, segment [First (Q), v] of Q, segment [u', w] of R, segment [v', Last (Q)] of Q, and segment [w', Finish] of R. (See Fig. 3.)

Case 2. I contains one half of B (and therefore $R - I$ contains at least one half of A). Find or establish the nonexistence of a triple (u, v, w) of vertices such that

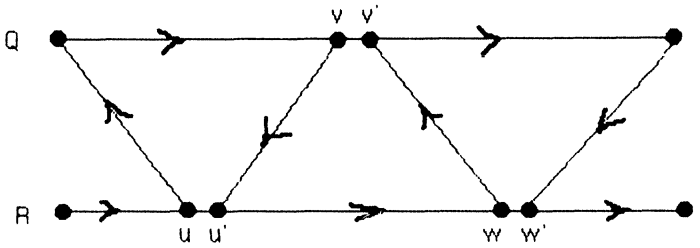

FIG. 3

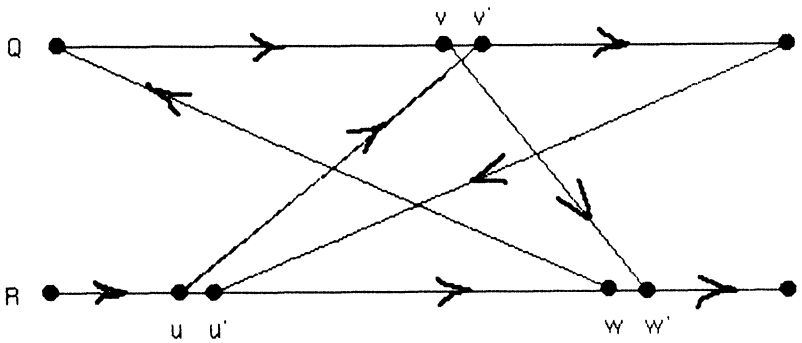

FIG. 4

$u' \in B \cap I$, v is a nonfinal vertex in Q, $w \in A \cap (R - I)$, and (u, v'), (v, w') are edges. In the negative case go to HPA3. In the positive case set R equal to the concatenation of segment [Start, u] of R, segment [v', Last (Q)] of Q, segment [u', w] of R, segment [First (Q), v] of Q, and segment [w', Finish] of R. (See Fig. 4.)

Theorem 2 of the Introduction is proved. □

REFERENCES

D. ANGLUIN AND L. G. VALIANT (1979), *Fast probabilistic algorithms for Hamiltonian circuits and matchings*, J. Comput. System Sci., 18, pp. 155–193.

R. BELLMAN (1960), *Combinatorial processes and dynamic programming*, in Proc. of the 10th Symposium in Appl. Math., Amer. Math. Soc., Providence, RI.

B. BOLLOBAS (1979), *Graph Theory*, Springer-Verlag, New York, Berlin.

B. BOLLOBAS, T. I. FENNER AND A. M. FRIEZE (1985), *An algorithm for finding Hamilton cycles in a random graph*, in Proc. 17th Annual ACM Symposium on Theory of Computing, pp. 430–439.

H. CHERNOFF (1952), *A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Ann. Math. Statist., 23, pp. 493–509.

P. ERDÖS AND J. SPENCER (1974), *Probabilistic Methods in Combinatorics*, Academic Press, New York.

M. R. GAREY AND D. S. JOHNSON (1979), *Computers and Intractability*, W. H. Freeman, San Francisco.

R. L. GRAHAM (1983), *Applications of the FGK inequality and its relatives*, in Mathematical Programming, The State of Art, Bonn 1982, Springer-Verlag, New York, Berlin.

Y. GUREVICH AND S. SHELAH (1984), *Expected computation time for Hamiltonian Path Problem and clique problem*, Tech. Report CRL-TR-50-84, Univ. of Michigan, Ann Arbor.

M. HELD AND R. M. KARP (1962), *A dynamic programming approach to sequencing problems*, J. Soc. Indust. Appl. Math., 10, pp. 196–210.

D. S. JOHNSON (1984), *The NP-completeness column: an ongoing guide*, J. Algorithms, 5, pp. 284–299.

L. LEVIN (1984), *Problems, complete in "average" instance*, in Proc. 16th Annual ACM Symposium on Theory of Computing, p. 465.

L. POSA (1976), *Hamiltonian circuits in random graphs*, Discrete Math., 14, pp. 359–364.

W. J. SAVITCH (1970), *Relationships between nondeterministic and deterministic tape complexities*, J. Comput. System Sci., 4, pp. 177–192.

E. SHAMIR (1983), *How many random edges make a graph Hamiltonian?* Combinatorica, 3, pp. 123–131.