JOURNAL OF COMPUTER AND SYSTEM SCIENCES 18, 155–193 (1979)

# Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings*

D. ANGLUIN AND L. G. VALIANT

*Computer Science Department, University of Edinburgh, Edinburgh, Scotland*

Received December 7, 1977; revised August 1, 1978

We describe and analyse three simple efficient algorithms with good probabilistic behaviour: two algorithms with run times of $O(n(\log n)^2)$ which almost certainly find directed (undirected) Hamiltonian circuits in random graphs of at least $cn \log n$ edges, and an algorithm with a run time of $O(n \log n)$ which almost certainly finds a perfect matching in a random graph of at least $cn \log n$ edges. Auxiliary propositions regarding conversion between input distributions and the "de-randomization" of randomized algorithms are proved. A new model, the random access computer (RAC), is introduced specifically to treat run times in low-level complexity.

## 1. INTRODUCTION

The main purpose of this paper is to give techniques for analysing the probabilistic performance of certain kinds of algorithms, and hence to suggest some fast algorithms with provably desirable probabilistic behaviour. The particular problems we consider are: finding Hamiltonian circuits in directed graphs (DHC), finding Hamiltonian circuits in undirected graphs (UHC), and finding perfect matchings in undirected graphs (PM). We show that for each problem there is a simple randomized algorithm which is extremely fast ($O(n(\log n)^2)$ for DHC and UHC, $O(n \log n)$ for PM) and which with probability tending to one finds a solution in randomly chosen graphs of sufficient density. These results contrast with the known [2, 15] *NP*-completeness of the first two problems (even for such dense inputs) and the best worst-case upper bound known of $O(n^{2.5})$ for the last [9].

We consider two different distributions for $n$ node random graphs following [8]: (i) $G_p$: graphs of $n$ nodes where each edge is present with probability $p$, independent of other edges, and (ii) $G_N$: graphs of $n$ nodes and exactly $N$ edges, each graph with equal probability. Conditions for the intertranslation of results for the two models are given in Section 3 which show that it suffices to prove our results in just one of the models.

* This research was supported by a grant from the Science Research Council. A preliminary version of this paper was presented at the Ninth ACM Symposium on Theory of Computing, Boulder, Colorado, U.S.A., May 2–4, 1977.



0022-0000/79/020155-39$02.00/0



571/18/2-4

The form of our results is the following: for any $n$, if $N \geqslant cn \log n$ then given a graph from $G_N$ the algorithm will find a solution with probability $1 - O(n^{-\alpha})$. Furthermore, $\alpha$ may be arbitrarily increased at the expense of increasing $c$ and the constant multiplicative factor of the run time. It is known that for any $\epsilon > 0$ if $N < (\frac{1}{2} - \epsilon)n \log n$ then $G_N$ contains isolated points with probability tending to one [6]. Since graphs with isolated points cannot contain Hamiltonian circuits or perfect matchings, it follows that the edge density $N$ required for our results cannot be improved by more than a constant factor.

We now summarize some of the previous work on these problems. Let $n$ be the number of nodes, let $Q$ be any function such that $Q(n) \to \infty$ as $n \to \infty$, and let $c$ be a (sufficiently large) constant. *Almost certainly* will mean "with probability $\to 1$ as $n \to \infty$."

Perepelica [22] presented $O(n^2)$ algorithms for almost certainly finding Hamiltonian circuits in directed and undirected graphs of at least $cn^{3/2}(\log n)^{1/2}$ edges. Pósa [23] showed that almost all undirected graphs of $cn \log n$ edges possess a Hamiltonian circuit; subsequently Komlós and Szemerédi [17] state that the required density may be reduced to $\frac{1}{2}(n \log n + n \log \log n + Q(n))$ edges. Pósa's proof does not rely explicitly on an algorithm, but various algorithms can be derived for finding undirected Hamiltonian circuits almost certainly, that are based on his central lemma. Independently, Koršunov [19] showed that a random undirected graph of at least $\frac{1}{2}(n \log + n \log \log n + Q(n))$ edges almost certainly has a Hamiltonian circuit, the proof being via the construction of an algorithm which almost certainly finds Hamiltonian circuits in such graphs. The algorithm described in [19] is for a density of $3n \log n$ edges and can be made to have a run time of $O(n^2(\log n)^2)$.

Concerning Hamiltonian circuits in directed graphs, the results appear to be fewer. Perepelica's $O(n^2)$ algorithm for density $cn^{3/2}(\log n)^{1/2}$ edges was mentioned above. Wright [28] proved that almost all directed graphs with $Q(n) \cdot n^{3/2}$ edges have a Hamiltonian circuit; his proof is nonalgorithmic.

For the existence of perfect matchings in undirected $n$-node graphs ($n$ even), Erdös and Rényi [7] showed that a random graph with $\frac{1}{2}n \log n + Q(n)$ edges almost certainly contains a perfect matching. Guimady and Perepelica [13] state the existence of an algorithm with a run-time of $O(n^2)$ which almost certainly finds a perfect matching in a graph from $G_p$ with $p \geqslant 2 \log n/n$ (i.e., about $n \log n$ edges).

Surveys of other probabilistic results in the field of efficient asymptotically exact algorithms for combinatorial problems may be found in Karp [16] and Guimady, Glebov and Perepelica [12].

Perhaps our main result is our $O(n(\log n)^2)$ algorithm for almost certainly finding a Hamiltonian circuit in a random directed graph of a $cn \log n$ edges. By an easy reduction, an algorithm for UHC of the same complexity follows. As a separate result we give a direct algorithm for UHC that has a similar complexity. For the PM problem we can improve the run-time to $O(n \log n)$; this result has as a consequence an $O(n \log n)$ expected time algorithm for determining (with certainty) the existence of PMs in such graphs.

These algorithmic results of course imply the almost certain existence of solutions in random graphs of the indicated density. In the case of DHC this existence result

appears to be new and does not follow from the result for UHC. The theorems of Wright [29] and Koršunov [18] allow one easily to transfer these results for labelled graphs to unlabelled graphs of the same density.

In comparing our results with previous ones the following generalizations appear to be valid: (i) Our three algorithms resemble "elegant heuristics" for the appropriate problems and no concession is made to complicate the algorithm in order to make the proofs easier. (ii) The run times are all $O(n(\log n)^2)$ as compared with at least $n^2$ for all corresponding previous ones. (iii) The proof technique consists of a set of simple lemmas that can be applied fairly systematically to a wide class of algorithms. Deriving the multiplicative factor in the threshold density is not our objective here. For that objective it is possible that some of the above three virtues would have to be sacrificed.

The reader will note that the algorithms as described in Section 3 are all *randomized* (i.e., make random decisions [24, 27]). As we show in Section 9, however, they can be translated into *deterministic* algorithms that simulate randomness by abstracting the necessary random bits from unused parts of the random input graph. This translation, while rather generally applicable, appears to be primarily of theoretical interest, since for practical use the randomized versions are preferable.

We wish to express the run times of our algorithms so as to reflect their performance on random access computers of the kind presently in common use. Because of the apparent absence from the literature of a model suitable to this purpose, we have chosen to define a new class of models. We believe that this may be a convenient one on which to standardize in general for expressing results in low-level complexity.

We are concerned here primarily with the computational problem of *efficiently finding* instances of solutions to the given problem. This kind of question appears in general to be more difficult than the problem of simply *determining the existence* of a solution. For example, for the clique problem in $G_p$ with $p = \frac{1}{2}$, it is known that a clique of size $(2 - \epsilon)\log_2 n$ almost certainly exists, but no polynomial time algorithm is known for almost certainly finding one [11]. In a non-probabilistic context this dichotomy between finding solutions and determining their existence is present in extreme forms in problems where the existence of a solution is certain and hence requires no computation. In some cases, of course, the proof of certainty does yield a fast algorithm (e.g., Dirac's proof [4] of the existence of an UHC in graphs where all nodes have degree at least $n/2$ implies an $O(n^2)$ algorithm for finding one.) In others, however, this is apparently not the case (e.g., the problem of finding the prime factorization of an $n$-digit integer.) In the case of *NP*-complete problems, such as DHC and UHC, it is known that the *worst-case* complexities of determining the existence of a solution and of finding one are polynomially related.

In conclusion we note the desirability and regrettable absence of available results in the probabilistic analysis of combinatorial problems, that relate not only to specific problems but have significant wider application. It is therefore perhaps worth mentioning that a technique introduced by Levin [21, 25] can be used to show that for a wide class of computational problems of the kind we consider here, and for most machine models, one can construct for each time complexity class an algorithm that has at least as good probabilistic behaviour as any algorithm in a slightly smaller complexity class.

## 2. Preliminaries

A *graph* $G$ is an ordered pair $(V, E)$ where $V$ is the set of *nodes* and $E$ is the set of *edges*. We will always assume that there are $n$ nodes labelled by $1, 2, \ldots, n$. $G$ is a *directed graph* if $E \subseteq \{(v, w) \mid v \neq w;\ v, w \in V\}$, where $(a, b)$ denotes the ordered pair of $a$ and $b$. $G$ is an *undirected graph* if $E \subseteq \{\{v, w\} \mid v \neq w;\ v, w \in V\}$. In either case we denote the cardinality of $E$ by $N$. We sometimes write $e \in G$ for $e \in E$.

A *Hamiltonian path* (*HP*) from $u$ to $v$ in a directed graph is a directed path from $u$ to $v$ that visits every node of $G$ exactly once. In the case $u = v$ such a path is called a *Hamiltonian circuit* (*HC*). Hamiltonian paths and circuits are defined analogously in the undirected case. A *perfect matching* (*PM*) in an undirected graph with $n = 2k$ nodes is a set of $k$ edges $e_1, e_2, \ldots, e_k$ such that $e_1 \cup e_2 \cup \cdots \cup e_k = V$.

Following [8] we introduce random variables $G_p$ and $G_N$ ($D_p$ and $D_N$) that range over undirected (directed) graphs with $n$ nodes. (Explicit dependence on $n$ is suppressed for readability and will be clear from context.) $G_p$ is defined by the property that $\Pr(\{u, v\} \in G_p) = p$ for each of the $\binom{n}{2}$ possible edges, mutually independently. $G_N$ is defined to take values that have exactly $N$ edges in such a way that all graphs with $n$ nodes and $N$ edges have the same probability. The definitions of $D_p$ and $D_N$ are identical to these except that in this case there are $n(n-1)$ possible edges $(u, v)$.

For brevity we shall frequently denote a finite sequence "$x_1, x_2, \ldots, x_i$" by "$x^i$" schematically. Unless otherwise indicated, all logarithms are to the base $e$.

We shall use the following inequalities:

Proposition 2.1. *For all real* $x$, $1 + x \leqslant e^x$.

Proposition 2.2. *For all integral* $n$, $k$ *with* $1 \leqslant k \leqslant n$

$$\binom{n}{k} \leqslant \left(\frac{ne}{k}\right)^k$$

Proposition 2.3. $\exists K > 0\ \forall n, p$ *with* $n > 0$, $1 > p > 0$, *and* $np$ *integral*

$$\binom{n}{np} p^{np}(1-p)^{n-np} \geqslant Kn^{-1/2}$$

Proposition 2.4. $\forall n, p, \beta$ *with* $0 \leqslant p \leqslant 1$, $0 \leqslant \beta \leqslant 1$

(a) $$\sum_{k=0}^{\lfloor (1-\beta)np \rfloor} \binom{n}{k} p^k (1-p)^{n-k} \leqslant \exp(-\beta^2 np/2)$$

(b) $$\sum_{k=\lceil (1+\beta)np \rceil}^{n} \binom{n}{k} p^k (1-p)^{n-k} \leqslant \exp(-\beta^2 np/3)$$

Propositions 2.2 and 2.3 follow from Stirling's approximation for the factorial function, and Proposition 2.4 from Chernoff's bound [8, p. 17].

To express the run times of our algorithms we define a model of computation called a *random access computer* (*RAC*). Details of the model and some comparisons with other models may be found in Appendix 1. Here we present an overview and motivation of the model for the convenience of the reader.

The problem addressed by the model is that of reconciling meaningfully the idea of *finite length words* with that of *asymptotic run times* expressed in terms of the input length $n$. The solution we propose is simply that the word length should be dependent on $n$.

An RAC is a fixed program machine. In any particular computation, its memory consists of a finite number of words all of the same finite length (in bits). RAC programs resemble programs for RAMS [3] except that they may involve a much richer set of instructions, e.g., $+$, $-$, $*$, $\div$, shifting, bitwise Boolean operations, jumping on zero, indirect addressing. Each such instruction is executed on a whole word (or words) and takes unit time. The input is assumed to be present in memory at the start of a computation, or equivalently, can be read into memory in unit time. For any instruction set we distinguish between a deterministic RAC and a *randomized RAC* (*r-RAC*) with that instruction set; the latter has an additional instruction which generates a wordful of "random bits."

The above suggests a family of models. For definiteness, we select one of these, which we describe in the Appendix. Essentially it has a set of standard arithmetic and Boolean instructions, and word length $k \log n$. The justification for the latter is that while we need at least $\log_2 n$ bits to address every word of the input, $k \log n$ is sufficient for any fixed polynomial space computation (for some constant $k$).

## 3. Input Distributions

It has been observed repeatedly in the literature on random graphs that most of the results that can be obtained for $G_p$ can also be obtained for $G_N$ and vice versa. In this section we give some sufficient conditions for translating results between $G_p(D_p)$ and $G_N(D_N)$ that are general enough to encompass all the applications we require. We also indicate a method for transferring certain kinds of results for $D_p$ to $G_p$.

A mapping $f$ from the set of all undirected graphs into the interval [0, 1] will be called a *graph function*. A mapping $f$ is a *graph predicate* iff $f$ is a graph function which takes no values other than 0 and 1. A graph function $f$ is *monotonic* iff whenever $G = (V, E)$, $G' = (V, E')$, and $E \subseteq E'$ then $f(G) \leqslant f(G')$.

For example:

$$f(G) = \begin{cases} 1 & \text{if } G \text{ has a HC} \\ 0 & \text{otherwise} \end{cases}$$

is a monotonic graph predicate, while

$$g(G) = \Pr(\text{UHC algorithm succeeds on } G)$$

is a graph function which is not in general monotonic.

Let $f$ be a graph function and $X$ be a random variable taking values in the set of undirected graphs. We let $s(f, X)$ denote the final outcome of the experiment of first drawing $G$ from $X$ and then drawing 1 with probability $f(G)$ and 0 with probability $1 - f(G)$.

Propositions 3.1 and 3.2 will show that the fixed $N$ and fixed $p$ models are inter-translatable even if monotonicity is not satisfied. Proposition 3.3 strengthens 3.2 under the assumption of monotonicity. Obvious analogues of these results hold for directed graphs.

PROPOSITION 3.1. *If $N(n)$ is any function such that $(\log n)/N(n) \to 0$ as $n \to \infty$ and if $f$ is any graph function such that*

$$N \geqslant N(n) \Rightarrow \Pr(s(f, G_N) = 1) = 1 - O(n^{-\alpha}) \tag{1}$$

*then*

$$p \geqslant p(n) \Rightarrow \Pr(s(f, G_p) = 1) = 1 - O(n^{-\alpha})$$

*if $p(n) = (N(n)/\binom{n}{2})(1 + 2\beta(n))$ and $\beta(n) = ((2\alpha \log n)/N(n))^{1/2}$.*

*Proof.* The conditions on $p$ imply that $N(n) \leqslant p\binom{n}{2}/(1 + 2\beta(n)) \leqslant (1 - \beta(n))\, p\binom{n}{2}$ for large enough $n$. Hence by Proposition 2.4(a),

$$\Pr(G_p \text{ has} \leqslant N(n) \text{ edges}) \leqslant \exp\left(-\tfrac{1}{2}(\beta(n))^2\, p \binom{n}{2}\right) \leqslant n^{-\alpha}. \tag{2}$$

Thus, abbreviating "$G_p$ has $N$ edges" by "$Q_N$":

$$\Pr(s(f, G_p) = 1) \geqslant \sum_{N \geqslant N(n)} \Pr(s(f, G_p) = 1 \mid Q_N) \cdot \Pr(Q_N) \tag{3}$$

and it is clear in general that

$$\Pr(s(f, G_p) = 1 \mid Q_N) = \Pr(s(f, G_N) = 1). \tag{4}$$

Substituting (1), (2), and (4) into (3) gives the desired result. ∎

PROPOSITION 3.2. *If $f$ is a graph function such that*

$$p \geqslant p(n) \Rightarrow \Pr(s(f, G_p) = 1) = 1 - O(n^{-\alpha}) \tag{5}$$

*then*

$$N \geqslant \binom{n}{2} p(n) \Rightarrow \Pr(s(f, G_N) = 1) = 1 - O(n^{1-\alpha}).$$

*Proof.* Given $N \geqslant \binom{n}{2} p(n)$, let $p = N/\binom{n}{2}$. By Proposition 2.3 there exists a constant $K > 0$, independent of $n$ and $N$, such that

$$\Pr(G_p \text{ has } N \text{ edges}) \geqslant K/n. \tag{6}$$

But from (4):

$$\Pr(s(f, G_N) = 0) \leqslant \Pr(s(f, G_p) = 0)/\Pr(G_p \text{ has } N \text{ edges})$$

Hence by (5) and (6)

$$\Pr(s(f, G_N) = 0) \leqslant (n/K)\, cn^{-\alpha} = O(n^{1-\alpha}). \quad \blacksquare$$

The loss of a factor of $n$ may be avoided if we consider only monotonic graph functions and restrict $N$ more. The following generalizes Theorem 3 of [23]:

PROPOSITION 3.3. *Let $f$ be a monotonic graph function and suppose that $p(n)$ is such that $(\log n)/(p(n)\binom{n}{2}) \to 0$ as $n \to \infty$. Then if*

$$p = p(n) \Rightarrow \Pr(s(f, G_p) = 1) = 1 - O(n^{-\alpha})$$

*then*

$$N \geqslant N(n) \Rightarrow \Pr(s(f, G_N) = 1) = 1 - O(n^{-\alpha})$$

*if $N(n) = \binom{n}{2} p(n)(1 + \beta(n))$ and $\beta(n) = ((3\alpha \log n)/(\binom{n}{2} p(n)))^{1/2}$.*

*Proof.* Define random variables $X$ and $Y$ as follows: select a graph $G = (V, E)$ according to $G_p$ and let $X = G$. If $G$ has more than $N$ edges then let $Y = *$. Otherwise, add $N - |E|$ edges to $G$ at random and let $Y$ take the resulting value.

From Proposition 2.4(b) and the assumptions about $N$, $\Pr(Y = *) = O(n^{-\alpha})$. By construction and the monotonicity of $f$:

$$\begin{aligned}\Pr(s(f, G_N) = 1) &= \Pr(s(f, Y) = 1 \mid Y \neq *) \\ &\geqslant \Pr(s(f, X) = 1 \mid Y \neq *).\end{aligned}$$

But

$$\begin{aligned}\Pr(s(f, X) = 1 \mid Y \neq *) &\geqslant 1 - \Pr(s(f, X) = 0) - \Pr(Y = *) \\ &= 1 - O(n^{-\alpha}). \quad \blacksquare\end{aligned}$$

Finally we observe that the following reduction may be used to derive from the DHC algorithm described in Section 4 an alternative algorithm for UHC. Note that this reduction assumes the availability of random elements for probabilities $p$ and $\frac{1}{2}$ (see Section 11 and Section 12).

PROPOFITION 3.4. *If we select $G$ according to $G_p$ and perform the following randomized procedure on it to produce $G'$, then $G'$ will be distributed according to $D_{p/2}$:*

(i) *if $\{i, j\}$ is an edge of $G$ then with probability*

$$\begin{aligned}\tfrac{1}{2} - p/4 &\text{ set } (i, j) \in G' \text{ and } (j, i) \notin G' \\ \tfrac{1}{2} - p/4 &\text{ set } (i, j) \notin G' \text{ and } (j, i) \in G' \\ p/4 &\text{ set } (i, j) \in G' \text{ and } (j, i) \in G' \\ p/4 &\text{ set } (i, j) \notin G' \text{ and } (j, i) \notin G'\end{aligned}$$

(ii) *if $\{i, j\}$ is not an edge of $G$, then set $(i, j) \notin G'$ and $(j, i) \notin G'$.*

*Proof.* It is easy to verify that for any pair $\{i, j\}$ $1 \leqslant i \neq j \leqslant n$ we have

$$\Pr((i,j) \in G' \text{ and } (j,i) \notin G') = p(1 - p/2)/2$$
$$\Pr((i,j) \notin G' \text{ and } (j,i) \in G') = p(1 - p/2)/2$$
$$\Pr((i,j) \in G' \text{ and } (j,i) \in G') = p^2/4$$
$$\Pr((i,j) \notin G' \text{ and } (j,i) \notin G') = (1 - p/2)^2.$$

Furthermore these probabilities are independent of the results for all other pairs. From this it may be deduced that for any directed graph $H$ with $N$ edges

$$\Pr(G' = H) = (p/2)^N(1 - p/2)^{n(n-1)-N}. \quad \blacksquare$$

Since any HC in the graph $G'$ constructed above can be made into a HC in the original $G$ by undirecting its edges, a randomized reduction from the UHC problem to the DHC problem is implied.

## 4. Algorithms and Main Results

In this section we give a high-level description of our three principal algorithms and state the main results about their probabilistic performance on inputs from $D_N$ or $G_N$. These algorithms all work equally well on $D_p$ or $G_p$ according to the translations presented in Section 3. The algorithms described are randomized. In Section 11 we show how to make them deterministic, and in Section 12 we analyse the run times of implementations of them on RACs and r-RACs.

The variable $G$ will initially be the graph which is presented as input to the algorithm, and will change during the course of the computation as edges are deleted from it. The algorithms all use the following procedure for selecting an edge randomly from a given node $u$ of $G$ and deleting it:

*procedure* SELECT($u$)
  *begin* *if* $\forall v \in V$ $(u, v) \notin G$ *then* return "$*$"
    *else* select at random with equal probabilities one of the edges $(u, v) \in G$, delete $(u, v)$ from $G$, and return the value $v$;
  *end*

(N.B. For the case of undirected graphs we replace $(u, v)$ by $\{u, v\}$ in the above.)

### *DHC Algorithm*

As input the algorithm takes a directed graph $G$ of $n$ nodes, together with two specified nodes $s$ and $t$ (which may be the same). The algorithm attempts to find a directed HP from $s$ to $t$ in $G$. If it succeeds, it returns "success," otherwise it returns "failure." During execution it maintains a *partial path* $P$ which consists either of a simple directed path, or of the disjoint union of a simple directed path and a simple directed cycle. In either case the terminal endpoint of the path is kept in a variable called ndp (for

*next departure point*) and the algorithm attempts to extend $P$ by calling SELECT to explore edges at random out of the node ndp.

Figure 1 depicts the nodes of $G$ and the value of $P$ at various stages in the algorithm.

*procedure* DHC($G$, $s$, $t$)
*begin*
1. ndp $\leftarrow s$, $P \leftarrow \varnothing$
2. (comment: $P$ consists of a directed path from $s$ to ndp)
   (a) If $P$ includes every node of $G$ (except $t$, in the case $s \neq t$) and if we previously explored and deleted (ndp, $t$) from $G$, then add (ndp, $t$) to $P$ and return "success."

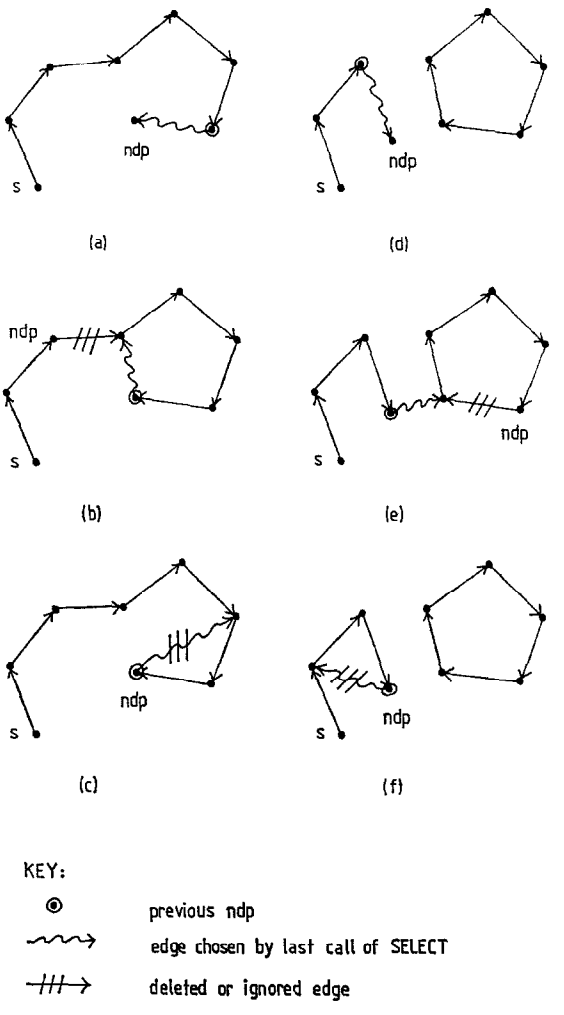


FIGURE 1

(b) $v \leftarrow$ SELECT(ndp)

(i) If $v =$ "$*$" then return "failure."

(ii) If $v \neq t$ and $v$ is not in $P$ then add (ndp, $v$) to $P$, ndp $\leftarrow v$, and go to 2 (Fig. 1(a)).

(iii) If $v \neq s$ and $v \in P$ and there are at least $n/2$ nodes in $P$ between $v$ and ndp (inclusive) then

*begin*
$u \leftarrow$ predecessor of $v$ in $P$,
delete $(u, v)$ from $P$,
add (ndp, $v$) to $P$,
ndp $\leftarrow u$,
go to 3
*end* (Fig. 1(b)).

(iv) Otherwise (i.e., if $v = s$ or $v = t$ or cycle is too small) go to 2. (Fig. 1(c))

3. (comment: $P$ consists of a directed path from $s$ to ndp, and a disjoint directed cycle of at least $n/2$ nodes.)

(a) $v \leftarrow$ SELECT(ndp)

(i) If $v =$ "$*$" then return "failure."

(ii) If $v \neq t$ and $v$ is not in $P$ then add (ndp, $v$) to $P$, ndp $\leftarrow v$, and go to 3. (Fig. 1(d))

(iii) If $v$ is in the cycle part of $P$ then

*begin*
$u \leftarrow$ predecessor of $v$ in $P$,
delete $(u, v)$ from $P$,
add (ndp, $v$) to $P$,
ndp $\leftarrow u$,
go to 2
*end* (Fig. 1(e)).

(iv) Otherwise go to 3 (Fig. 1(f)).

*end*

DHC THEOREM 4.1. $\forall\alpha\ \exists M, c\ \forall N \geqslant cn \log n\ \forall s, t\ (1 \leqslant s,\ t \leqslant n)$ *the probability that* DHC($D_N$, $s$, $t$) *returns "success" before* SELECT *has been called* $Mn \log n$ *times is* $1 - O(n^{-\alpha})$.

This theorem is proved in Section 6 and Section 7. In Section 11 and Section 12 an $O(n(\log n)^2)$ time bound is deduced for an implementation on a RAC.

*PM Algorithm*

The algorithm takes as input an undirected graph $G$ of $n$ nodes, where $n$ is an even integer. It attempts to find a perfect matching in $G$, returning "success" if it succeeds, and "failure" otherwise. It maintains a *partial matching* $P$ (i.e., a perfect matching

on a subset of the nodes) and tries randomly to augment it by exploring new edges until it either fails for lack of edges or succeeds in finding a PM.

```
procedure PM(G)
  begin
    1. P ← ∅
    2. (a) If P includes all the nodes of G, then return "success"
       (b) Otherwise, ndp ← least-numbered node not in P, and go to 3
    3. v ← SELECT(ndp)
       (a) If v = "*" then return "failure"
       (b) If v is not in P then add {ndp, v} to P and go to 2
       (c) If v is in P then
           begin
             u ← unique node such that {u, v} ∈ P,
             delete {u, v} from P,
             add {ndp, v} to P,
             ndp ← u,
             go to 3
           end
  end
```

PM THEOREM 4.2. $\forall\alpha\ \exists M, c\ \forall N \geqslant cn \log n$ *the probability that* PM($G_N$) *returns "success" before* SELECT *has been called* $Mn \log n$ *times is* $1 - O(n^{-\alpha})$.

The proof of this theorem is given in Sections 6, 8, 9. In Section 11 and Section 12 time bounds are deduced of $O(n \log n)$ on a r-RAC and $O(n(\log n)^2)$ on a RAC.

COROLLARY 4.3. *There is an algorithm* PM′ *with the following properties:*

(i) *Given any undirected graph* PM′ *outputs a perfect matching if one exists and "failure" otherwise.*

(ii) $\exists K, c\ \forall n, N \geqslant cn \log n$ *the expected run time of* PM′($G_N$) *is less than* $Kn \log n$ *on a* r-RAC, $Kn(\log n)^2$ *on a* RAC.

*Proof.* PM′ consists of first running PM and if this fails, then running a worst-case $O(n^k)$ time algorithm for the same problem (e.g., [5, 9]). If $K$ and $c$ are chosen large enough that the probability of failure of PM is $O(n^{-\alpha})$ where $\alpha \geqslant k - 1$ then PM′ will have the claimed expected run time. ∎

*UHC Algorithm*

As input the algorithm takes an undirected graph $G$ of $n$ nodes and two specified nodes $s$ and $t$ (which may be the same). It attempts to find a HP from $s$ to $t$, returning "success" if it succeeds and "failure" otherwise. It maintains a *partial path* $P$ which consists of a simple path with $s$ as one endpoint. The other endpoint is kept in the variable

ndp. The algorithm attempts to extend $P$ by calling SELECT to explore edges at random out of ndp.

*procedure* UHC($G$, $s$, $t$)
*begin*
1. ndp $\leftarrow s$, $P \leftarrow \varnothing$.
2. (comment: $P$ consists of a path from $s$ to ndp.)
   (a) If $P$ includes every node of $G$ (except $t$, in the case $s \neq t$), and if we previously deleted {ndp, $t$} from $G$, then we add {ndp, $t$} to $P$ and return "success."
   (b) $v \leftarrow$ SELECT(ndp)
      (i) If $v =$ "$*$" then return "failure."
      (ii) If $v \neq t$ and $v$ is not in $P$ then add {ndp, $v$} to $P$, ndp $\leftarrow v$, and go to 2.
      (iii) If $v \neq t$ and $v$ is in $P$ then
      *begin*
        $u \leftarrow$ the neighbour of $v$ in $P$ which is closer to ndp,
        delete $\{u, v\}$ from $P$,
        add {ndp, $v$} to $P$,
        ndp $\leftarrow u$,
        go to 2
      *end*
      (iv) Otherwise (i.e., if $v = t$) go to 2.
*end*

UHC THEOREM 4.4. $\forall\alpha\ \exists M, c\ \forall N \geqslant cn \log n\ \forall s, t\ (1 \leqslant s,\ t \leqslant n)$ *the probability that* UHC($G_N$, $s$, $t$) *returns "success" before* SELECT *has been called* $Mn \log n$ *times is* $1 - O(n^{-\alpha})$.

The proof of this theorem is indicated in Sections 6, 8, 10. In Section 11 and Section 12 an $O(n(\log n)^2)$ implementation is deduced for a RAC.

## 5. SUMMARY OF PROOFS

The basic form of all three algorithms in Section 4 is similar: they start at some node, jump at random to one of the neighbours of the node (deleting the edge used), accordingly adjust some data-structure that determines which node is to be jumped from next, then jump at random to one of the neighbours of this node, and so on, until all the nodes have been visited in this way (possibly with some further, similar, processing to form a path or cycle). If instead at each step we simply selected at random and equiprobably one of the $n$ nodes of the graph then the result of the "Coupon Collector's Problem" [10, p. 225] says we should expect to make $n \log n$ selections in order to visit every node. This is essentially the phenomenon exploited in the proofs of the DHC, PM, and UHC Theorems.

Below we give informal outlines of the proofs using the same variables as appear in the formal proofs.

*Outline of DHC Proof*

At each choice by DHC, every node not hitherto arrived at from the current departure point is equally likely to be arrived at next (EP Lemma, 7.1). The probability that a given node is not arrived at during $K_2 n \log n$ choices made by DHC is thus $\leqslant (1 - 1/n)^{K_2 n \log n}$ or $O(n^{-K_2})$. Thus, for $K_2$ sufficiently large, the probability that DHC runs for $K_2 n \log n$ choices and fails to visit every node is negligible (B Lemma, 7.4).

Given $M \geqslant K_2$ what is the probability that DHC fails before making $Mn \log n$ choices? This happens only if DHC exhausts all of the edges leaving some node. For any fixed $K$ we may neglect graphs in which some node has fewer than $K \log n$ edges leaving it by taking the density sufficiently large (Sociability Lemma, 6.1). So we would like to show that the probability that DHC departs from some node more than $K \log n$ times in the first $Mn \log n$ choices is negligible. We do this as follows:

For a given node, DHC's departures from it fall into groups; in each group, DHC first makes the node the departure point, and then repeatedly makes choices from the node until some choice allows DHC to "escape" to a new departure point. Since each choice has a probability of about $\frac{1}{2}$ of permitting DHC to "escape," we would expect that the number of choices made from a given node to be about twice the number of times that node is made the departure point. The probability that this expected value is much exceeded is negligible (J Sublemma, 7.8).

A given node is made the departure point only when DHC arrives at it or at a specific neighbour of it (as determined by the current value of the partial path). Provided that no node has had very many choices made from it, each such arrival has probability bounded by about $1/n$. Thus in $Mn \log n$ choices we expect a given node to be made the departure point about $M \log n$ times, and we show that the probability that this expected value is much exceeded is negligible (H Sublemma, 7.7).

Combining this bound on the number of times a node is made the departure point with the bound on the ratio between the number of choices made from a node and the number of times it is made the departure point, we get a bound of $K_1 \log n$ for the number of choices made from any node (A Lemma, 7.6). Taking $K$ large with respect to $K_1$, we may neglect the probability that DHC exhausts the edges leaving some node before making $Mn \log n$ choices.

The remaining possibility for how DHC may fail is that it may run for $Mn \log n$ choices, having arrived at every node by the $(K_2 n \log n)$th choice, but without, in the $(M - K_2)n \log n$ remaining choices, being able to "close" the path via an edge from the departure point in a "path" state to the designated endpoint $t$.

The probability of remaining in the "path and cycle" state is about $\frac{1}{2}$ at each choice, so we expect about half the remaining $(M - K_2)n \log n$ choices to be made from departure points in the "path" state; the probability that this expected value much exceeds the actual value is negligible. Thus $K_3 n \log n$ choices from the "path" state may be guaranteed, for any fixed $K_3$, by choosing $M$ large enough with respect to $K_2$ and $K_3$ (C Lemma, 7.5).

At each choice in the "path" state either DHC terminates successfully because it has previously found an edge between the departure point and $t$, or it arrives at $t$ (which causes successful termination at the next choice), or it fails to arrive at $t$. Failure to arrive at $t$ has a probability of $\leqslant(1 - 1/n)$, so for the $K_3 n \log n$ choices guaranteed from the "path" state, the probability that DHC fails to close the path is $\leqslant(1 - 1/n)^{K_3 n \log n} = O(n^{-K_3})$, which is negligible for $K_3$ sufficiently large (D Lemma, 7.3).

*Outline of PM and UHC Proofs*

We first use the Sociability Lemma, 6.1, to neglect those graphs in which the degree of any node is less than $K \log n$ ($K$ a fixed constant to be determined later).

The Almost Equiprobability Lemma, 8.2, states that if an expedition has not used more than some fixed fraction of $K \log n$ edges at any node, then every node not hitherto arrived at from the current departure point is *almost* equally likely to be arrived at next.

Using the fact that each departure in PM or UHC from a given node must be preceded by an arrival at a specific destination (either the given node or a particular neighbour of it, depending on the value of the current partial matching or partial path), and that each such arrival has probability about $1/n$, we show that the probability that PM or UHC visits any node more than $K_1 \log n$ times within the first $Mn \log n$ choices is negligible for $K_1$ chosen large enough with respect to $M$, and $K$ with respect to $K_1$ (Lemmas 9.2, 10.2).

For PM it then suffices to note that for $M$ chosen large enough the probability that PM runs for $Mn \log n$ choices without visiting every node is negligible, since visiting every node gives a perfect matching (Lemma 9.3).

For UHC we show not only that it is highly probable that every node is visited within the first $K_2 n \log n$ choices (Lemma 10.3), but also that the algorithm then continues running long enough ($(M - K_2)n \log n$ choices) to terminate the path successfully at node $t$ (Lemma 10.4).

## 6. Basic Lemmas

We call $v$ a *neighbour* of $u$ in $G$ if $G$ is undirected and $\{u, v\}$ is an edge of $G$ or if $G$ is directed and $(u, v)$ is an edge of $G$. Assuming a particular value of $n$ given by context we define $S(d) = \{G \mid G$ is an $n$ node graph in which every node has at least $d$ neighbours$\}$. The implied graphs are directed or undirected according to context. Also, we denote either of the events $G_N \in S(K \log n)$ or $D_N \in S(K \log n)$ by $Z(K)$.

Sociability Lemma 6.1. $\forall \alpha, K\ \exists c\ \forall N \geqslant cn \log n$

$$\Pr(Z(K)) = 1 - O(n^{-\alpha})$$

*for both $G_N$ and $D_N$.*

*Proof.* By Proposition 3.2 it suffices to prove this result for $G_p$ with $p \geqslant (c \log n)/n$. Given $\alpha$, $K$ we let $c = 2(\alpha + K + 1)$. If $v$ is any node, the probability that $v$ has fewer than $K \log n$ neighbours can be bounded by $O(n^{-\alpha-1})$ by applying Proposition 2.4(a). The result follows by summing over the $n$ choices of $v$. ∎

The following result on conditional probabilities we shall also use widely. It can be verified by induction on $m$ and $k$.

BOTTLENECK LEMMA 6.2. *Suppose $X_1, X_2, \ldots, X_m$ are random variables taking values in a finite set $S$. Let $Q$ be any event. Let*

$$U = \{s^m \mid s_i \in S \text{ for } i = 1, 2, \ldots, m\}$$

*Suppose $Y \subseteq U$ has the property that there exist probabilities $p_1, p_2, \ldots, p_k$ such that for each $y^m \in Y$ there exist integers $i_1 < i_2 < \cdots < i_k$ such that*

$$\Pr(X_{i_j} \in C_Y(y^{i_j-1}) \mid X^{i_j-1} = y^{i_j-1} \wedge Q) \leqslant p_j$$

*where*

$$C_Y(z^i) = \{s \in S \mid \exists y^m \in Y \text{ with } y^i = z^i \text{ and } y_{i+1} = s\}$$

*(that is, $C_Y(z^i)$ is the set of continuations of $z^i$ that keep it in $Y$). Then*

$$\Pr(X^m \in Y \mid Q) \leqslant p_1 \cdot p_2 \cdot \cdots \cdot p_k .$$

This result may be visualized with the aid of a rooted tree of depth $m$ and uniform $|S|$-way branching. A path from the root to a node at depth $d$ represents in the obvious fashion a possible outcome of drawing values for $X_1, X_2, \ldots, X_d$. The set $Y$ corresponds to a set of paths from the root to certain of the leaves of the tree; we will imagine all of the nodes in these paths to be coloured green. The condition on $Y$ specifies that along each path of $Y$ there exist $k$ nodes ("bottlenecks") such that at the $i$th such node the conditional probability of drawing a green successor is at most $p_i$.

*Note.* Here, as in the remainder of the paper, we could avoid conditioning on the empty event by adding an appropriate further hypothesis. To avoid unnecessary complication we have omitted such hypotheses throughout; they may be supplied without difficulty by the reader.

## 7. PROOF OF THE DHC THEOREM

The values of $n$, $N$, $s$, $t$ will be fixed by context; we shall omit unnecessary references to them. We consider the experiment of drawing a graph from the distribution $D_N$ and then running the DHC algorithm on this graph. The random variable $G$ will denote the graph chosen, and the random variables $T_1, T_2, \ldots, T_{n(n-1)}$ will record the random selections of edges so that $T_i = *$ if DHC returns before calling SELECT $i$ times, and $T_i =$ the value returned by the $i$th call of SELECT otherwise.

We define $w^k$ to be an *expedition* iff either $k = 0$ or $1 \leqslant k \leqslant n(n-1)$ and $\Pr(T^k = w^k) > 0$. If $w^k$ is an expedition, then we may simulate DHC up to the $(k+1)$th call of SELECT (or return, if this occurs earlier) using $w_i$ as the value returned by the $i$th call of SELECT, even without knowing the value of $G$. In this context, $w^k$ will be called *finished* if DHC returns before the $(k+1)$th call of SELECT, otherwise it will be called *unfinished*. If $w^k$ is unfinished then $P(w^k)$ and $ndp(w^k)$ will be the values of the variables $P$ and ndp at the $(k+1)$th call of SELECT, and, if $\Pr(G = H \wedge T^k = w^k) > 0$, $G(H, w^k)$ will be the value of the graph $G$ at the $(k+1)$th call of SELECT if $G = H$ initially.

We note that if $\Pr(G = H \wedge T^k = w^k) > 0$ then in consequence of the random selection of edges

$$\Pr(T_{k+1} = w_{k+1} \mid G = H \wedge T^k = w^k) = \begin{cases} 1/d & \text{if } w^k \text{ is unfinished and } w_{k+1} \text{ is one of the } d > 0 \text{ neighbours of } \mathrm{ndp}(w^k) \text{ in } G(H, w^k) \\ 1 & \text{if } w_{k+1} = * \text{ and either } w^k \text{ is finished or } \mathrm{ndp}(w^k) \text{ has no neighbours in } G(H, w^k) \\ 0 & \text{otherwise.} \end{cases}$$

Let $w^k$ be an expedition. For each $j$, $1 \leqslant j \leqslant k$, let $v_j = \mathrm{ndp}(w^{j-1})$ if $w^{j-1}$ is unfinished, and $v_j = *$ otherwise. For $v \in V$ and $i = 1, 2, \ldots, k$ we have the following definitions:

(i) $(v_i, w_i)$ is the $i$th *step of* $w^k$

(ii) $w^k$ *departs from* $v$ *at step* $i$ iff $v = v_i$

(iii) $w^k$ *arrives at* $v$ *at step* $i$ iff $v = w_i$

(iv) $w^k$ *makes* $v$ *the departure point at step* $(i-1)$ iff either $i = 1$ and $v = v_1$ or $i > 1$ and $v = v_i$ while $v \neq v_{i-1}$. ("step 0" is a slight anomaly in this phrasing.)

(v) $w^k$ *has a chance to close at step* $i$ iff $w^{i-1}$ is unfinished and $P(w^{i-1})$ consists of a simple directed path containing all the nodes (except $t$ in the case $s \neq t$).

(vi) $w^k$ *returns "success"* (*"failure"*) iff DHC simulated with $w^k$ returns "success" ("failure") before calling SELECT $k+1$ times.

We define $w^k$ *departs from* $v$ $m$ *times* iff $m = |\{i \mid w^k \text{ departs from } v \text{ at step } i\}|$ and similarly for (iii), (iv), (v). We say also: $Q(w^k)$ *within the first* $m$ *steps* iff $Q(w^m)$, where $m \leqslant k$, and $Q$ is an appropriate predicate (e.g. returns "success.")

We next give a fundamental lemma that states that the random selection of edges by DHC makes each possible continuation of an expedition that has not exhausted the edges out of its next departure point equally likely.

EQUIPROBABILITY (EP) LEMMA 7.1. *Let* $d$ *be a positive integer such that* $dn \leqslant N$. *Suppose that* $w^k$ *is an unfinished expedition with* $u = \mathrm{ndp}(w^k)$ *and suppose that* $w^k$ *departs*

*from u exactly b times where $b < d$. Then for any node z such that $(u, z)$ is not a step of $w^k$*

$$\Pr(T_{k+1} = z \mid T^k = w^k \wedge G \in S(d)) = 1/(n - b - 1).$$

*Proof.* By symmetry. (N.B. In contrast to the undirected case, where conditioning problems arise (see Section 8), the result here is immediate.) ▮

For simplicity we abbreviate $T^{n(n-1)}$ and $w^{n(n-1)}$ to $T$ and $w$ in the remainder of this section. We define for any $M > 0$

$$F_M = \{w \mid w \text{ is an expedition which does not return success within the first } Mn \log n \text{ steps}\}.$$

The DHC Theorem, 3.1, is then equivalent to the following:

THEOREM 7.2. $\forall \alpha\, \exists c, M\, \forall N \geqslant cn \log n$

$$\Pr(T \in F_M) = O(n^{-\alpha}).$$

*Proof.* First note that for all $K_1$

$$\Pr(T \in F_M) \leqslant (1 - \Pr(Z(K_1))) + \Pr(T \in F_M \mid Z(K_1)).$$

The Sociability Lemma 6.1, can be used to bound the first term. To bound the second we introduce the following sets of expeditions, which are defined in terms of $n$, $M$, $K_1$, $K_2$, $K_3$ (here regarded as general variables):

$A = \{w \mid w^i$ departs from some node at least $K_1 \log n$ times, where $i = Mn \log n\}$

$B = \{w \mid w \notin A$ and $w^i$ does not arrive at every node other than $s$, where $i = K_2 n \log n\}$

$C = \{w \mid w \notin A,\ w \notin B,\ w \in F_M$ and $w$ has fewer than $K_3 n \log n$ chances to close within the first $Mn \log n$ steps$\}$

$D = F_M - (A \cup B \cup C)$.

Clearly $F_M \subseteq A \cup B \cup C \cup D$ for all $n$, $M$, $K_1$, $K_2$, $K_3$ and therefore

$$\begin{aligned}\Pr(T \in F_M \mid Z(K_1)) \leqslant \Pr(T \in A \mid Z(K_1)) \\ + \Pr(T \in B \mid Z(K_1)) \\ + \Pr(T \in C \mid Z(K_1)) \\ + \Pr(T \in D \mid Z(K_1)).\end{aligned}$$

For each of the quantities on the right hand side we prove a lemma to bound it:

D LEMMA 7.3. $\forall \alpha\, \exists K_3\, \forall K_1, K_2\, \forall M \geqslant K_3$

$$\Pr(T \in D \mid Z(K_1)) = O(n^{-\alpha}).$$

B LEMMA 7.4. $\forall\alpha\ \exists K_2\ \forall K_1\ \forall M \geqslant K_2$

$$\Pr(T \in B \mid Z(K_1)) = O(n^{-\alpha}).$$

Hence for any $\alpha$ we can find $K_2$ and $K_3$ satisfying the above two lemmas simultaneously, for all $K_1$ and for all $M \geqslant K_2, K_3$.

C LEMMA 7.5. $\forall\alpha, K_2, K_3\ \exists M \geqslant \max(K_2, K_3)\ \forall K_1$

$$\Pr(T \in C \mid Z(K_1)) = O(n^{-\alpha}).$$

Hence we can find a suitable $M(\alpha, K_2, K_3)$ also.

A LEMMA 7.6. $\forall\alpha, M\ \exists K_1$

$$\Pr(T \in A \mid Z(K_1)) = O(n^{-\alpha}).$$

Hence we can find an appropriate $K_1(\alpha, M)$. Finally, by the Sociability Lemma 6.1, we can choose a $c(\alpha, K_1)$ such that $N \geqslant cn \log n$ guarantees that

$$\Pr(Z(K_1)) = 1 - O(n^{-\alpha}).$$

This establishes the theorem. It remains to prove the four lemmas. ∎

*Proof of* A *Lemma* 7.6. Define the following two sets of expeditions:

$H = \{w \mid$ for some $i \leqslant Mn \log n$, $w^i$ departs from each node $< K_1 \log n$ times, and makes some node the departure point $\geqslant K \log n$ times$\}$

$J = \{w \mid$ for some $i \leqslant Mn \log n$, $w^i$ makes each node the departure point $\leqslant K \log n$ times and departs from some node $\geqslant K_1 \log n$ times$\}$.

The reader can easily verify that for all $K$, $K_1$, and $M$, $A \subseteq (H \cup J)$. It therefore suffices to prove the following results:

H SUBLEMMA 7.7. $\forall\alpha, M\ \exists K\ \forall K_1$

$$\Pr(T \in H \mid Z(K_1)) = O(n^{-\alpha}).$$

J SUBLEMMA 7.8. $\forall\alpha, M, K\ \exists K_1$

$$\Pr(T \in J \mid Z(K_1)) = O(n^{-\alpha}).$$

Clearly for all $\alpha$ and $M$ we can find a $K(\alpha, M)$ by the H Sublemma and hence a $K_1(\alpha, M)$ by the J Sublemma such that both are satisfied simultaneously. The A Lemma therefore follows. ∎

*Proof of* H *Sublemma* 7.7. Let $X = \{Y \mid Y \subseteq \{1,\ldots, Mn \log n\}$ and $\mid Y \mid = K \log n\}$. For $v \in V - \{s\}$ and $Y \in X$ define

$$H(v, Y) = \{w \in H \mid \text{for each } i \in Y,\ w \text{ makes } v \text{ the departure point at step } i,\ \text{and } w^i \text{ departs from each node } < K_1 \log n \text{ times}\}.$$

Then

$$H = \bigcup_{\substack{v \in V \\ Y \in X}} H(v, Y)$$

and hence

$$\Pr(T \in H \mid Z(K_1)) \leqslant \sum_{\substack{v \in V \\ Y \in X}} \Pr(T \in H(v, Y) \mid Z(K_1))$$

Fix $v, Y, w \in H(v, Y)$, and $i \in Y$. Clearly $w^{i-1}$ is unfinished and there are two possibilities:

(i) Assume $v \notin P(w^{i-1})$. Then if $v$ is made the departure point at step $i$, $w$ must arrive at $v$ at step $i$ (i.e., $T_i = v$). Hence, by the EP Lemma, 7.1,

$$\Pr(T_i = v \mid T^{i-1} = w^{i-1} \wedge Z(K_1)) \leqslant 1/(n - K_1 \log n - 1).$$

(ii) Assume $v \in P(w^{i-1})$. Then there is a unique node $y$ such that $(v, y) \in P(w^{i-1})$. and if $v$ is to be made the departure point at step $i$, then it must be that $T_i = y$. Hence by the EP Lemma 7.1,

$$\Pr(T_i = y \mid T^{i-1} = w^{i-1} \wedge Z(K_1)) \leqslant 1/(n - K_1 \log n - 1).$$

In either of these cases, the probability of a continuation of $w^{i-1}$ which remains in $H(v, Y)$ is at most $1/(n - K_1 \log n - 1)$. Since this holds for every $i \in Y$ it follows by the Bottleneck Lemma 6.2, that

$$\Pr(T \in H(v, Y) \mid Z(K_1)) \leqslant (n - K_1 \log n - 1)^{-K \log n}$$

Summing over all $v \in V$ and $Y \in X$ and appealing to Proposition 2.2 gives

$$\begin{aligned}\Pr(T \in H \mid Z(K_1)) &\leqslant n \binom{Mn \log n}{K \log n} (n - K_1 \log n - 1)^{-K \log n} \\ &\leqslant n \left(\frac{Me(1 + o(1))}{K} h(K_1)\right)^{K \log n}\end{aligned}$$

where $h(K_1) \to 1$ as $n \to \infty$ for any fixed $K_1$. Choosing $K > \max\{\alpha + 1, Me^2\}$ gives the result. ∎

*Proof of J Sublemma* 7.8. Let

$$X = \Big\{\langle m_1, \ldots, m_r\rangle \mid 1 \leqslant r \leqslant K \log n, \text{ each } m_i \text{ is a positive integer, and } \sum m_i = \lceil K_1 \log n \rceil\Big\}.$$

For any $v \in V$ and $Y \in X$ let

$$J(v, Y) = \{w \in J \mid Y = \langle m_1, \ldots, m_r\rangle \text{ and there exist } s_1, \ldots, s_r \text{ such that for each } j = 1, 2, \ldots, r, \ w \text{ makes } v \text{ the departure point at step } s_j \text{ and departs from } v \text{ at steps } s_j + 1, \ldots, s_j + m_j. \text{ Furthermore, } w^i \text{ departs from each node } < K_1 \log n \text{ times, where } i = s_r + m_r - 1\}.$$

Clearly

$$J = \bigcup_{\substack{v \in V \\ Y \in X}} J(v, Y).$$

Fix $v \in V$, $Y \in X$ and $w \in J(v, Y)$. Suppose $Y = \langle m_1, \ldots, m_r\rangle$. The first $K_1 \log n$ departures from $v$ appear in $r$ groups:

$$s_1 + 1, \ldots, s_1 + m_1 \ ; \ s_2 + 1, \ldots, s_2 + m_2 \ ; \cdots ; \ s_r + 1, \ldots, s_r + m_r \ ;$$

Fix $j$ with $1 \leqslant j \leqslant r$ and assume that $m_j > 1$. Let $i$ be such that $s_j + 1 \leqslant i < s_j + m_j$. Then $w^{i-1}$ is unfinished and departs from each node fewer than $K_1 \log n$ times. There are two cases:

(i) Suppose $P(w^{i-1})$ consists of a simple path from $s$ to $v$. Let

$$F = \{y \mid y = t \text{ or } y \text{ is in } P(w^{i-1}) \text{ and there are fewer than } n/2 \text{ nodes between } y \text{ and } v \text{ in } P(w^{i-1})\}.$$

Clearly $|F| \leqslant n/2 + 1$ and $w^{i-1}$ will depart again from $v$ at step $i + 1$ (which it must do to remain in $J(v, Y)$) iff $T_i \in F$. By the EP Lemma 7.1,

$$\Pr(T_i \in F \mid T^{i-1} = w^{i-1} \wedge Z(K_1)) \leqslant \frac{(1 + n/2)}{(n - K_1 \log n - 1)}.$$

(ii) Suppose $P(w^{i-1})$ consists of a path from $s$ to $v$ and a cycle of $\geqslant n/2$ nodes. Let

$$F = \{y \mid y = t \text{ or } y \text{ is in the path from } s \text{ to } v\}.$$

Clearly $|F| \leqslant 1 + n/2$ and $w^{i-1}$ will depart again from $v$ at step $i + 1$ only if $T_i \in F$. By the EP Lemma 7.1,

$$\Pr(T_i \in F \mid T^{i-1} = w^{i-1} \wedge Z(K_1)) \leqslant \frac{(1 + n/2)}{(n - K_1 \log n - 1)}.$$

Thus, in either case, the probability of a continuation of $w^{i-1}$ which remains in $J(v, Y)$ is bounded by $h(K_1)(1 + o(1))/2$ where $h(K_1) \to 1$ as $n \to \infty$ for any fixed $K_1$. Since

there are $\geqslant (K_1 - K)\log n$ distinct values of $i$ for which this holds, it follows by the Bottleneck Lemma 6.2, that

$$\Pr(T \in J(v, Y) \mid Z(K_1)) \leqslant (h(K_1)(1 + o(1))/2)^{(K_1-K)\log n}$$

Summing over $v \in V$ and $Y \in X$ and noting that provided $K_1 \geqslant 2K$,

$$|X| \leqslant K \log n \binom{K_1 \log n}{K \log n}$$

and that this can be bounded by Proposition 2.2, we have

$$\Pr(T \in J \mid Z(K_1)) \leqslant Kn \log n \left(\frac{2K_1 e}{K}\right)^{K \log n} \left(\frac{1 + o(1)}{2} h(K_1)\right)^{K_1 \log n}$$

This will be $O(n^{-\alpha})$ provided $K_1$ is chosen large enough with respect to $\alpha$, $K$. ∎

*Proof of* B *Lemma* 7.4. Given $\alpha$ choose $K_2 > \alpha + 1$, and suppose that $M \geqslant K_2$ and $K_1$ is arbitrary. For each $v \in V$, $v \neq s$, let

$$B(v) = \{w \in B \mid v \text{ is the least numbered node other than } s \text{ such that } w^i \text{ does not arrive at } v, \text{ where } i = K_2 n \log n\}.$$

Clearly

$$B = \bigcup_{v \in V} B(v).$$

Fix $v \in V$ $(v \neq s)$ and $w \in B(v)$. For $1 \leqslant i \leqslant K_2 n \log n$, $w^i$ clearly cannot return "success," since it has not visited $v$. Also, we may ignore the possibility that $w^i$ returns "failure," for in that case $w \notin A$ and $M \geqslant K_2$ imply that $\Pr(T = w \mid Z(K_1)) = 0$. Therefore we may assume $w^{i-1}$ is unfinished and hence by the EP Lemma 7.1,

$$\Pr(T_i \neq v \mid T^{i-1} = w^{i-1} \wedge Z(K_1)) \leqslant 1 - 1/n.$$

Thus the probability of a continuation of $w^{i-1}$ remaining in $B(v)$ is $\leqslant 1 - 1/n$ for each $i$, and hence, by the Bottleneck Lemma 6.2,

$$\Pr(T \in B(v) \mid Z(K_1)) \leqslant (1 - 1/n)^{K_2 n \log n}$$

Summing over $v$ and using Proposition 2.1 gives

$$\Pr(T \in B \mid Z(K_1)) \leqslant ne^{-K_2 \log n} = O(n^{-\alpha}). \quad ∎$$

*Proof of* C *Lemma* 7.5. Let

$$X = \{Y \subseteq \{1, 2, \ldots, (M - K_2)n \log n\} \mid |Y| = K_3 n \log n\}.$$

For $Y \in X$ let

$$C(Y) = \{w \in C \mid \text{for each } i \text{ with } 1 \leqslant i \leqslant (M - K_2)n \log n, \text{ if } w \text{ has a chance to close at step } (i + K_2 n \log n) \text{ then } i \in Y\}.$$

Clearly

$$C = \bigcup_{Y \in X} C(Y)$$

Fix $Y \in X$ and $w \in C(Y)$. Let

$$W = \{1, 2, \ldots, (M - K_2)n \log n\} - Y$$

and suppose that $(i + 1 - K_2 n \log n) \in W$. Since $C \subseteq F_M$, $w^{i+1}$ does not return "success." We may assume that $w^{i+1}$ does not return "failure," because $w \notin A$ implies that $\Pr(T = w \mid Z(K_1)) = 0$. Since $w \notin B$, we know that $w^i$ arrives at every node other than $s$. By our choice of $i$, $w$ does not have a chance to close at step $i + 1$, which implies that $P(w^i)$ consists of a path and a cycle. To analyse the probability of $P(w^i)$ consisting of a path and a cycle we consider the two possible cases:

(i) Assume that $P(w^{i-1})$ consists of a simple path. If $F$ is the set of all nodes $y$ in $P(w^{i-1})$ such that there are $\geqslant n/2$ nodes between $y$ and $\mathrm{ndp}(w^{i-1})$ then it follows from the EP Lemma, 7.1, that

$$\Pr(T_i \in F \mid T^{i-1} = w^{i-1} \wedge Z(K_1)) \leqslant (n/2)/(n - K_1 \log n - 1).$$

(ii) Assume that $P(w^{i-1})$ consists of a path and a cycle $\geqslant n/2$ nodes. If $F$ is the set of all the nodes that are not in the cycle then it follows from the EP Lemma 7.1, that

$$\Pr(T_i \in F \mid T^{i-1} = w^{i-1} \wedge Z(K_1)) \leqslant (n/2)/(n - K_1 \log n - 1).$$

Thus in either case the probability of a continuation remaining in $C(Y)$ is at most $h(K_1)(1 + o(1))/2$ (where $h(K_1) \to 1$ as $n \to \infty$ for any $K_1 > 0$), for each of the $(M - K_2 - K_3)n \log n$ values of $i$ such that $((i + 1) - K_2 n \log n) \in W$. By applying the Bottleneck Lemma 6.2, summing over $Y \in X$, and applying Proposition 2.2,

$$\begin{aligned}\Pr(T \in C \mid Z(K_1)) &\leqslant \binom{(M - K_2)\, n \log n}{K_3 n \log n} \cdot \left(h(K_1) \cdot \frac{1 + o(1)}{2}\right)^{(M-K_2-K_3)n \log n} \\ &\leqslant \left(\frac{2(M - K_2)\, e}{K_3}\right)^{K_3 n \log n} \cdot \left(h(K_1) \cdot \frac{1 + o(1)}{2}\right)^{(M-K_2)n \log n}\end{aligned}$$

which is $O(n^{-\alpha})$ for $M$ sufficiently large with respect to $\alpha$, $K_2$, and $K_3$, for any $K_1$. ∎

*Proof of* D *Lemma* 7.3. Let $\alpha$ be given and choose $K_3 \geqslant \alpha$. Let $K_1$, $K_2$, and $M \geqslant K_3$ be given. Suppose $w \in D$. Then $w \notin C$ ensures that for some set $\{i_1, \ldots, i_r\}$ such that $1 \leqslant i_1 < i_2 \cdots < i_r \leqslant Mn \log n$ where $r = K_3 n \log n$, $w$ has a chance to close at each step $i_j$. Fix $m = i_j$ for some $1 \leqslant j \leqslant r$. Since $w \in F_M - A$, we need consider only the case when $w^{m-1}$ is unfinished. If $u = \mathrm{ndp}(w^{m-1})$ then $(u, t)$ cannot be a step of $w^m$ for otherwise $w^m$ would return "success." Then by the EP Lemma 7.1,

$$\Pr(T_m \neq t \mid T^{m-1} = w^{m-1} \wedge Z(K_1)) \leqslant 1 - 1/n$$

and by the Bottleneck Lemma 6.2, and Proposition 2.1:

$$\Pr(T \in D \mid Z(K_1)) \leqslant (1 - 1/n)^{K_3 n \log n} = O(n^{-\alpha})$$

by our choice of $K_3$. ∎

## 8. Basic Lemmas for Undirected Analysis

We first modify the definitions of Section 7 for the case of undirected graphs, and then prove the Almost Equiprobability (AEP) Lemma, which serves as the analogue of the EP Lemma in the undirected case.

The EP Lemma 7.1, does not hold in the case of undirected graphs, since edges can be explored in either direction, so that the results of explorations from $v$ will in general condition the probability of arriving at $v$.

We consider the experiment of drawing a graph from the distribution $G_N$ and then running the algorithm PM (UHC) on it. The definitions and remarks in Section 7 up to the EP Lemma 7.1, then hold with the following changes:

(a) $G_N$ for $D_N$

(b) $\binom{n}{2}$ for $n(n-1)$

(c) PM (UHC) for DHC

(d) items (iv) and (v) do not apply.

Further, we now say that the expedition $w^k$ *visits* $v$ at step $i$ iff either $w^k$ departs from $v$ at step $i$ or $w^k$ arrives at $v$ at step $i$.

The following two lemmas hold for both the PM and the UHC algorithm, and also for any other similar algorithm in which (i) new edges are explored by calling SELECT from an ndp, and (ii) at each step the ndp is determined uniquely by the expedition and independent of the graph.

The first lemma bounds the decrease in probability of a particular expedition in a given graph when the degree of a node is increased by one. (Straightforward generalizations to larger perturbations can be derived easily.)

Perturbation Lemma 8.1. *Suppose that $G_1$ and $G_2$ are graphs, that $w^k$ is a possible expedition on both $G_1$ and $G_2$, and that for some node $v$,*

$$\text{degree}(v \text{ in } G_2) = 1 + \text{degree}(v \text{ in } G_1) = 1 + d,$$

*and* $\text{degree}(u \text{ in } G_2) \leqslant \text{degree}(u \text{ in } G_1)\ \forall u \neq v$, *and that $w^k$ visits $v$ at most $b$ times. Then*

$$\Pr(T^k = w^k \mid G = G_1) \leqslant K \cdot \Pr(T^k = w^k \mid G = G_2)$$

*where* $K = \exp(b/(d - b + 1))$.

*Proof.* The initial relations between the degrees of nodes of $G_1$ and $G_2$ are preserved throughout the expedition, since the same edges are explored and deleted in both graphs. If $w^{i-1}$ is unfinished and $\text{ndp}(w^{i-1}) = v$ and the degree of $v$ in $G(G_1, w^{i-1})$ is $f$ then

$$\Pr(T_i = w_i \mid T^{i-1} = w^{i-1} \wedge G = G_1) = 1/f$$

and

$$\Pr(T_i = w_i \mid T^{i-1} = w^{i-1} \wedge G = G_2) = 1/(f+1)$$

while for any other value of $\text{ndp}(w^{i-1})$

$$\Pr(T_i = w_i \mid T^{i-1} = w^{i-1} \wedge G = G_1) \leqslant \Pr(T_i = w_i \mid T^{i-1} = w^{i-1} \wedge G = G_2).$$

Hence we have

$$\Pr(T^k = w^k \mid G = G_1) = \frac{1}{d_1} \cdots \frac{1}{d_r} Q_1$$

$$\Pr(T^k = w^k \mid G = G_2) = \frac{1}{d_1 + 1} \cdots \frac{1}{d_r + 1} Q_2$$

where $w^k$ departs from $v$ $r$ times and $d_1, \ldots, d_r$ are the degrees of $v$ in $G_1$ at the successive departures, and $Q_1 \leqslant Q_2$. Thus the ratio of the two probabilities is bounded by

$$(1 + 1/d_1) \cdots (1 + 1/d_r) \leqslant \exp(r/d_r).$$

Since $r \leqslant b$ and $d_r \geqslant d - b + 1$ the result follows. ∎

The proof makes it evident that the result still holds when the hypothesis degree($v$ in $G_2$) = 1 + degree($v$ in $G_1$) is replaced by degree($v$ in $G_2$) $\leqslant$ 1 + degree($v$ in $G_1$).

ALMOST EQUIPROBABILITY (AEP) LEMMA 8.2. *Suppose $w^k$ is an unfinished expedition, with $u = \text{ndp}(w^k)$. Suppose $w^k$ visits each node at most $r \log n$ times and $z$ is a node such that $z \neq u$, and neither $(u, z)$ nor $(z, u)$ is a step of $w^k$. Then*

$$\frac{1}{(1+\epsilon)\, n} \leqslant \Pr(T_{k+1} = z \mid T^k = w^k \wedge Z(K)) \leqslant \frac{1+\epsilon}{n - r \log n - 1}$$

*if $K > r$, $c \geqslant 2K$, $N \geqslant cn \log n$ and $n$ is sufficiently large, where*

$$1 + \epsilon = (c/(c - K)) \exp(r/(K - r)).$$

*Proof.* Let $z_1$ and $z_2$ be any two of the at least $n - r \log n - 1$ possible arrival points (i.e. nodes satisfying the hypotheses on $z$ above). We must bound the relative probabilities of arriving at $z_1$ and $z_2$. Let

$$R = 1/(\Pr(T^k = w^k \wedge Z(K)))$$

and

$$g(x, H) = \Pr(T_{k+1} = x \wedge T^k = w^k \wedge G = H).$$

Then for $i = 1, 2$

$$\Pr(T_{k+1} = z_i \mid T^k = w^k \wedge Z(K)) = R \sum_{H \in A_i} g(z_i, H) \tag{1}$$

where

$$A_i = \{H \in S(K \log n) \mid \Pr(T^k = w^k \wedge G = H) > 0 \text{ and } \{u, z_i\} \text{ is an edge of } H\}.$$

Let $S_1 = A_1 - A_2$, $S_2 = A_2 - A_1$, $S_3 = A_1 \cap A_2$. Clearly, for all $H \in S_3$, $g(z_1, H) = g(z_2, H)$ and hence

$$\sum_{H \in S_3} g(z_1, H) = \sum_{H \in S_3} g(z_2, H) \tag{2}$$

Let

$$B_1 = \{H \in S_1 \mid \text{degree}(z_1) \text{ in } H \text{ is exactly } K \log n\}$$

and

$$C_1 = S_1 - B_1.$$

Each member $H$ of $C_1$ can be mapped to a distinct member $H^*$ of $S_2$ by replacing the edge $\{u, z_1\}$ by the edge $\{u, z_2\}$. Since this mapping $H \mapsto H^*$ satisfies the hypotheses of the Perturbation Lemma 8.1, then for each $H \in C_1$

$$\Pr(T^k = w^k \mid G = H) \leqslant K_1 \Pr(T^k = w^k \mid G = H^*)$$

where $K_1 = \exp(r/(K - r))$. Also for each $H \in C_1$, $g(z_1, H) = g(z_2, H^*)$. So

$$\sum_{H \in C_1} g(z_1, H) \leqslant K_1 \sum_{H \in S_2} g(z_2, H). \tag{3}$$

It remains to bound the contributions of $H \in B_1$. For each $H \in B_1$ we define:

$$Q(H) = \{(v, x) \mid \{v, x\} \in H;\ (v, x) \text{ and } (x, v) \text{ are not steps of } w^k;\ \text{degree}(x) \text{ in } H \geqslant K \log n + 1;\ v \neq z_1, z_2;\ \{v, z_1\} \notin H\}$$

$$F(H) = \{J \mid J \text{ can be obtained from } H \text{ by choosing some } (v, x) \in Q(H) \text{ and then replacing } \{u, z_1\} \text{ by } \{u, z_2\} \text{ and replacing } \{v, x\} \text{ by } \{v, z_1\} \text{ in } H\}.$$

Let $H \in B_1$ and $J \in F(H)$ be arbitrary. Clearly $J \in S_2$ and the degree of just one node, namely $z_2$, may be larger (by at most one) in $J$ than in $H$. Thus

$$\Pr(T_{k+1} = z_1 \mid T^k = w^k \wedge G = H) \leqslant \Pr(T_{k+1} = z_2 \mid T^k = w^k \wedge G = J)$$

and also $H$ and $J$ satisfy the hypotheses of the Perturbation Lemma 8.1, so

$$\Pr(T^k = w^k \mid G = H) \leqslant K_1 \Pr(T^k = w^k \mid G = J)$$

where $K_1$ was defined above to be $\exp(r/K - r)$. From these two inequalities we may conclude

$$g(z_1, H) \leqslant K_1 g(z_2, J).$$

We shall now show that for all $H \in B_1$, $J \in S_2$ and large enough $n$,

$$|F(H)| \geqslant (c - K)n \log n - K^2 \log^2 n \tag{4}$$

and

$$|F^{-1}(J)| \leqslant (n - K \log n)K \log n. \tag{5}$$

It will then follow that

$$\begin{aligned} &((c - K)\, n \log n - K^2 \log^2 n) \sum_{H \in B_1} g(z_1, H) \\ &\leqslant K_1 \sum_{H \in B_1} \sum_{J \in F(H)} g(z_2, J) \\ &\leqslant K_1(n - K \log n)\, K \log n \sum_{J \in S_2} g(z_2, J). \end{aligned} \tag{6}$$

*Proof of* (4). $w^k$ uses fewer than $K \log n$ edges from each node. Hence $K \log n$ edges can be set aside at each node, including all those explored by $w^k$, and still at least $(c - K)n \log n$ edges remain. Of these, at most $\frac{1}{2}(K \log n + 2)^2$ can have both endpoints in the set $\{z_1, z_2, \text{neighbours of } z_1 \text{ in } H\}$. This gives a lower bound for $|Q(H)|$. Since clearly $|Q(H)| = |F(H)|$, result (4) follows.

*Proof of* (5). The only way of recovering from $J \in F(B_1)$ a value $H \in F^{-1}(J)$ is the following:

(i) select one of the $K \log n$ nodes $v$ such that $\{v, z_1\} \in J$

(ii) select one of the at most $n - K \log n$ nodes $x$ which is not a neighbour of $v$ in $J$

(iii) replace $\{u, z_2\}$ by $\{u, z_1\}$ and $\{v, z_1\}$ by $\{v, x\}$.

We now combine results (2), (3), and (6) and use the facts that $A_2 = S_3 \cup S_2$ and $A_1 = S_3 \cup C_1 \cup B_1$ to give

$$\sum_{H \in A_1} g(z_1, H) \leqslant K_1(c/(c - K)) \sum_{H \in A_2} g(z_2, H).$$

Applying (1) shows that all the (at least $(n - r \log n - 1)$) possible arrival points have probabilities differing from each other by a factor of at most $1 + \epsilon$. The result follows. ∎

## 9. Proof of PM Theorem

$T$ and $w$ will denote $T^{\binom{n}{2}}$ and $w^{\binom{n}{2}}$ throughout. For any $M > 0$ define

$$F_M = \{w \mid w \text{ is an expedition which does not return "success" within the first } Mn \log n \text{ steps}\}.$$

The PM Theorem 4.2, is then equivalent to

THEOREM 9.1. $\forall\alpha\ \exists M, c\ \forall N \geqslant cn \log n$

$$\Pr(T \in F_M) = O(n^{-\alpha}).$$

*Proof.* For any $K$ we have

$$\Pr(T \in F_M) \leqslant (1 - \Pr(Z(K))) + \Pr(T \in F_M \mid Z(K)). \tag{1}$$

For any $M$ and $K_1$ define

$$A = \{w \mid w \text{ is an expedition such that } w^i \text{ visits some node } \geqslant K_1 \log n \text{ times, where } i = Mn \log n\}$$

$$B = \{w \mid w \text{ is an expedition, } w \notin A, \text{ and } w^i \text{ fails to visit some node, where } i = Mn \log n\}.$$

Once the algorithm visits a node, the node remains in the partial matching unless the algorithm exhausts all of the edges incident to the node and returns "failure." If and when the algorithm visits the last remaining unvisited node for the first time, it completes a perfect matching and returns "success." Thus $F_M \subseteq A \cup B$ for any $M$ and $K_1$. Hence for any $M$, $K$, $K_1$:

$$\Pr(T \in F_M \mid Z(K)) \leqslant \Pr(T \in A \mid Z(K)) + \Pr(T \in B \mid Z(K)). \tag{2}$$

Below we shall prove the following two Lemmas:

LEMMA 9.2. $\forall\alpha,\ M\ \exists K_1, K', r_1\ \forall K \geqslant K',\ N \geqslant r_1 Kn \log n$

$$\Pr(T \in A \mid Z(K)) = O(n^{-\alpha}). \tag{3}$$

LEMMA 9.3. $\forall\alpha\ \exists M\ \forall K_1\ \exists K'', r_2\ \forall K \geqslant K'',\ N \geqslant r_2 Kn \log n$

$$\Pr(T \in B \mid Z(K)) = O(n^{-\alpha}). \tag{4}$$

Thus, given $\alpha$, we may choose $M$ to satisfy Lemma 9.3. For this $\alpha$ and $M$ we may choose $K_1$, $K'$, $r_1$ to satisfy Lemma 9.2. For the specified $\alpha$ and $M$ and this $K_1$ we may choose $K''$, $r_2$ to satisfy Lemma 9.3. We then choose $K \geqslant K', K''$. The Sociability Lemma 6.1, guarantees that we may choose $c \geqslant r_1 K, r_2 K$ so that $N \geqslant cn \log n$ implies $\Pr(Z(K)) = 1 - O(n^{-\alpha})$. Combining this with (1), (2), (3), and (4) we conclude that for the chosen values of $M$ and $c$, and for all $N \geqslant cn \log n$, $\Pr(T \in F_M) = O(n^{-\alpha})$. ∎

*Proof of Lemma* 9.2. Let $\alpha$ and $M$ be given. Fix $K_1$ and decompose $A$ according to the first $K_1 \log n$ "too frequent" visits as follows: Let

$$X = \{Y \subseteq \{1, 2, \ldots, Mn \log n\} \mid \mid Y \mid = K_1 \log n\}$$

$$D = \{0, 1\}^{K_1 \log n}$$

For each $v \in V$, $Y \in X$, $d \in D$ let

$$A(v, Y, d) = \{w \in A \mid \text{for each } i \in Y,\ w^i \text{ visits each node at most } K_1 \log n \text{ times};\ w \text{ visits } v \text{ at step } i, \text{ and the visit is an arrival if } d_j = 0, \text{ and a departure if } d_j = 1, \text{ where } i \text{ is the } j\text{th smallest element of } Y\}.$$

Then

$$A = \bigcup_{v,Y,d} A(v, Y, d).$$

Fix $v \in V$, $Y \in X$, $d \in D$, $w \in A(v, Y, d)$ and $i \in Y$, where $i$ is the $j$th smallest element of $Y$ for some $j > 1$. We consider the two possible cases for $d_j$:

(i) $d_j = 0$. Then $w$ must arrive at $v$ at step $i$. By the AEP Lemma 8.2,

$$\Pr(T_i = v \mid T^{i-1} = w^{i-1} \wedge Z(K)) \leqslant \frac{1+\epsilon}{n - K_1 \log n - 1}$$

provided $K > K_1$, $c \geqslant 2K$, $N \geqslant cn \log n$, where

$$1 + \epsilon = (c/(c-K)) \exp(K_1/(K - K_1)).$$

(ii) $d_j = 1$. Then $w$ must depart from $v$ at step $i$. Since $j > 1$, $w$ visits $v$ before step $i$. In order to depart from a previously visited node, PM must arrive at the mate of the node in the current partial matching at the preceding step. That is, there is a unique node $z \neq v$ such that $\{v, z\} \in P(w^{i-2})$ and $z$ is the only continuation of $w^{i-2}$ which remains in $A(v, Y, d)$. Applying the AEP Lemma 8.2,

$$\Pr(T_{i-1} = z \mid T^{i-2} = w^{i-2} \wedge Z(K)) \leqslant \frac{1+\epsilon}{n - K_1 \log n - 1}$$

under the same provisos as in case (i).

Since the node $z$ in case (ii) is distinct from $v$, we have shown that for at least $K_1 \log n - 1$ distinct values of $i$, the probability of a continuation of $w^i$ which remains in $A(v, Y, d)$ is at most $(1+\epsilon)/(n - K_1 \log n - 1)$. Applying the Bottleneck Lemma 6.2:

$$\Pr(T \in A(v, Y, d) \mid Z(K)) \leqslant \left(\frac{1+\epsilon}{n - K_1 \log n - 1}\right)^{K_1 \log n - 1}$$

Summing over $v$, $Y$, $d$ and applying Proposition 2.2:

$$\begin{aligned}\Pr(T \in A \mid Z(K)) &\leqslant n \binom{Mn \log n}{K_1 \log n} 2^{K_1 \log n} \left(\frac{1+\epsilon}{n - K_1 \log n - 1}\right)^{K_1 \log n - 1} \\ &\leqslant n^2 h(K_1) \left(\frac{2Mc \exp(K/(K-K_1))}{(c-K)K_1}\right)^{K_1 \log n}\end{aligned}$$

where $h(K_1) \to 1$ as $n \to \infty$ for any fixed $K_1$. This last bound will be $O(n^{-\alpha})$ for $K_1$ chosen sufficiently large with respect to $M$, $K$ with respect to $K_1$, and $c$ with respect to $K$, for all $N$ exceeding $cn \log n$. ∎

*Proof of Lemma* 9.3. Let $\alpha$ be given. Fix $M$ and $K_1$. For each $v \in V$ define

$$B(v) = \{w \in B \mid w^i \text{ fails to visit } v, \text{ where } i = Mn \log n\}.$$

Clearly

$$B = \bigcup_{v \in V} B(v).$$

Let $v \in V$, $w \in B(v)$ and consider any $i$ with $1 \leqslant i \leqslant Mn \log n$. Since $w \in B$, $w^{i-1}$ cannot return "success." If $w^{i-1}$ returns "failure," then $\Pr(T = w \mid Z(K)) = 0$ provided $K > K_1$, because $w \notin A$ implies that $w^{i-1}$ visits no node more than $K_1 \log n$ times. Thus we may assume $w^{i-1}$ is unfinished. If $u = \text{ndp}(w^{i-1})$ then neither $(u, v)$ nor $(v, u)$ can be a step of $w^{i-1}$, so by the AEP Lemma 8.2, we have:

$$\Pr(T_i \neq v \mid T^{i-1} = w^{i-1} \wedge Z(K)) \leqslant \left(1 - \frac{1}{(1+\epsilon)\, n}\right)$$

provided $K \geqslant K_1$, $c \geqslant 2K_1$, $N \geqslant cn \log n$, where

$$1 + \epsilon = (c/(c - K)) \exp(K_1/(K - K_1)).$$

Applying the Bottleneck Lemma 6.2, summing over $v$, and applying Proposition 2.1:

$$\begin{aligned}\Pr(T \in B \mid Z(K)) &\leqslant n\left(1 - \frac{1}{(1+\epsilon)\, n}\right)^{Mn \log n} \\ &\leqslant \exp((1 - M/(1+\epsilon)) \cdot \log n)\end{aligned}$$

which will be $O(n^{-\alpha})$ provided $M$ is sufficiently large with respect to $\alpha$, for any $K_1$, provided $K$ is sufficiently large with respect to $K_1$, and $c$ with respect to $K$, for all $N \geqslant cn \log n$. ∎

## 10. Outline of Proof of UHC Theorem

In this section we give the decomposition Lemmas for the proof of the UHC Theorem; the Lemmas may be proved by further decompositions and case arguments as in Section 9. We denote $T_2^{(n)}$ and $w_2^{(n)}$ by $T$ and $w$ throughout this section. For any $M > 0$ define:

$$F_M = \{w \mid w \text{ is an expedition which does not return "success" within the first } Mn \log n \text{ steps}\}.$$

The UHC Theorem 4.4, is reformulated as:

Theorem 10.1. $\forall \alpha\ \exists M, c\ \forall N \geqslant cn \log n$

$$\Pr(T \in F_M) = O(n^{-\alpha}).$$

*Proof.* For any $K$

$$\Pr(T \in F_M) \leqslant (1 - \Pr(Z(K))) + \Pr(T \in F_M \mid Z(K)). \tag{1}$$

For any $M$, $K_1$, $K_2$ define:

$$A = \{w \mid w \text{ is an expedition and } w^i \text{ visits some node more than } K_1 \log n \text{ times, where } i = Mn \log n\}$$

$$B = \{w \mid w \text{ is an expedition, } w \notin A, \text{ and } w^i \text{ does not visit every node, where } i = K_2 n \log n\}$$

$$C = F_M - (A \cup B).$$

Clearly

$$\begin{aligned}\Pr(T \in F_M \mid Z(K)) \leqslant \Pr(T \in A \mid Z(K)) \\ + \Pr(T \in B \mid Z(K)) \\ + \Pr(T \in C \mid Z(K)).\end{aligned} \tag{2}$$

Lemmas bounding each of these quantities are stated below; their proofs are omitted.

LEMMA 10.2. $\forall \alpha, M\ \exists K_1, K', r_1\ \forall K \geqslant K', N \geqslant r_1 K n \log n$

$$\Pr(T \in A \mid Z(K)) = O(n^{-\alpha}). \tag{3}$$

LEMMA 10.3. $\forall \alpha\ \exists K_2\ \forall K_1\ \exists K'', r_2\ \forall K \geqslant K'', M \geqslant K_2, \quad N \geqslant r_2 K n \log n$

$$\Pr(T \in B \mid Z(K)) = O(n^{-\alpha}). \tag{4}$$

LEMMA 10.4. $\forall \alpha, K_2\ \exists M \geqslant K_2\ \forall K_1\ \exists K''', r_3\ \forall K \geqslant K''', \quad N \geqslant r_3 K n \log n$

$$\Pr(T \in C \mid Z(K)) = O(n^{-\alpha}). \tag{5}$$

Thus, given $\alpha$ we choose $K_2$ to satisfy 10.3. For this $\alpha$ and $K_2$ we choose $M \geqslant K_2$ to satisfy 10.4. For $\alpha$ and $M$ we choose $K_1$, $K'$, $r_1$ to satisfy 10.2. For $\alpha$, $K_2$, $M$, $K_1$ we then choose $K''$ and $r_2$ to satisfy 10.3. We choose $K \geqslant K'$, $K''$, $K'''$ and, using the Sociability Lemma 6.1, we choose a value of $c \geqslant r_1 K, r_2 K, r_3 K$ such that $N \geqslant cn \log n$ implies that $\Pr(Z(K)) = 1 - O(n^{-\alpha})$. For these values of $K$, $M$, and $c$ we combine (1), (2), (3), (4), and (5) to show that $N \geqslant cn \log n$ implies $\Pr(T \in F_M) = O(n^{-\alpha})$. ∎

## 11. DERANDOMIZATION

The foregoing analysis was facilitated by the fact that we were dealing with randomized algorithms. The purpose of the present section is to show that each of the main algorithms can in principle be "derandomized" so as to become a deterministic algorithm. The

idea is simply to use some otherwise unused parts of the random input graph to provide the necessary random decisions in the algorithm. This derandomization process appears to serve only the theoretical purpose of making direct comparisons with deterministic algorithms possible.

We shall discuss derandomization of the DHC algorithm for input distribution $D_p$ with $(c \log n)/n \leqslant p \leqslant 1 - (c \log n)/n$. (The case of the remaining, extremely dense, graphs will be discussed at the end of this section.) The same method can be easily applied to the PM and UHC algorithms. By invoking Propositions 3.1 and 3.2 the derandomized results for all three problems for $D_N$ or $G_N$ follow immediately.

The deterministic DHC procedure works as follows. Let

$$L = \{1, 2, \ldots, \lfloor n/2 \rfloor\}$$
$$U = \{\lfloor n/2 \rfloor + 1, \ldots, n\}$$
$$t_1 = \max\{j \in L \mid j \neq 1 \text{ and } (j, n) \in G\}$$
$$t_2 = \min\{j \in U \mid j \neq n \text{ and } (j, 1) \in G\}.$$

(The algorithm returns "failure" in the unlikely event that either of these latter two sets is empty.) Simulate DHC twice, once on the subgraph of $G$ induced by $L$, with $s = 1$ and $t = t_1$, and once on the subgraph of $G$ induced by $U$, with $s = n$ and $t = t_2$. The success of both of these clearly yields a HC in $G$. Edges of $G$ going between $L' = L - \{1\}$ and $U' = U - \{n\}$ are independent of the two induced subgraphs; we indicate how to use these edges as a source of the random bits required in the two simulations of DHC.

Considering the pairs in $L' \times U'$ we can regard their appearance or non-appearance as edges of $G$ as at least $kn^2$ probability $p$ Bernoulli trials ($k$ a fixed positive constant arbitrarily near to $\frac{1}{4}$). If we divide these trials up into $pkn^2$ segments of $1/p$ bits each, then with high probability at least some fixed fraction $\gamma > 0$ of the segments will contain exactly one 1, and the position of the 1 within the segment will provide about $\log_2(1/p)$ random bits (i.e., probability $\frac{1}{2}$ Bernoulli trials). Hence at least $\gamma kcn(\log n)^2$ random bits can be expected, and it is highly probable that this will be enough for $\epsilon n \log n$ edge selections, provided $c$ is large with respect to $\epsilon$. (We note that if $p = (c \log n)/n$ then $O(n \log n \log \log n)$ bits suffice.) This description applies in the case $p \leqslant \frac{1}{2}$. In the alternative case, one proceeds similarly, but looking for 0's instead of 1's.

We formalize these claims in the following Lemmas, which may be verified using Propositions 2.2 and 2.4.

The first Lemma ensures that the algorithm can be specified without knowing $p$. It suffices to use an approximation $\hat{p}$ which is the ratio of the number of edges of $G$ in $U' \times L'$ to the cardinality of $U' \times L'$. (Note that the edges in $U' \times L'$ are independent of those in $L' \times U'$.)

LEMMA 11.1. $\forall \alpha\ \exists d > 0$

$$\Pr(|\, p - \hat{p}\, |) \geqslant d(p \log n)^{1/2}/n = O(n^{-\alpha}).$$

We choose the segments of the source to be of length $\lfloor 1/\hat{p} \rfloor$.

LEMMA 11.2. $\exists \gamma > 0\ \forall d$

$$|p - \hat{p}| < d(p \log n)^{1/2}/n \Rightarrow \Pr(\textit{any given segment has exactly one } 1) > \gamma$$

*for all sufficiently large n.*

LEMMA 11.3. $\exists \delta > 0\ \forall \alpha, d$

$$|p - \hat{p}| < d(p \log n)^{1/2}/n \Rightarrow \Pr(\textit{fewer than } \delta n^2 \hat{p} \log_2(1/\hat{p}) \textit{ bits can be generated from the source by the method described}) = O(n^{-\alpha}).$$

Since the quantity $\hat{p} \log_2(1/\hat{p})$ is minimized for $\hat{p} \in [(c \log n)/n - d(p \log n)^{1/2}/n, \frac{1}{2} + d(p \log n)^{1/2}/n]$ at the lower endpoint, for $n$ sufficiently large the claimed $\delta' cn(\log n)^2$ random bits can be expected with appropriate probability, for some fixed $\delta' > 0$. We leave to Section 12 the task of analyzing the complexity of extracting these random bits.

The only case we have not considered is that of very dense graphs. If $p > 1 - (c \log n)/n$ then we may not have enough "randomness" in the input graph to apply the above method. However, at this density nearly any sensible deterministic method will work with overwhelming probability, since we can afford to test for the presence of named edges ("$(u, v) \in G$?") with little probability of failure. It is not difficult to show that there is a simple procedure which will find a HC with probability $1 - O(n^{-\alpha})$ in $D_p$, provided $p(n)$ is bounded below by some positive constant, and which can be implemented to run in time $O(n \log n)$ on a RAC for adjacency-list representation of the input graph.

## 12. IMPLEMENTATION

We shall now outline possible implementations of the three main algorithms, first for r-RACs, and then, with the aid of derandomization, for RACs. We emphasize that for practical purposes we would recommend the randomized versions of the algorithms in conjunction with a random number generator. The deterministic implementations are included only for the theoretical purpose of making direct comparisons with other deterministic algorithms possible.

### *Representation of Inputs*

In the directed case we assume that $G = (V, E)$ is given in adjacency list form. Each list $L_v = \{w \mid (v, w) \in E\}$ is stored in increasing order in the RAC memory. Also the value of $n = |V|$ is available in a word, and there are two length $n$ vectors SA[$v$] and LEN[$v$] giving respectively the start address and length of $L_v$ for $v = 1, 2, \ldots, n$.

In the undirected case the representation is similar, with lists $L_v = \{w \mid \{v, w\} \in E\}$. However the two copies of $\{v, w\}$ (i.e., in $L_v$ and $L_w$) are doubly-linked so that they can be easily deleted together.

Whenever an edge $(v, w)$ is deleted the entry for $w$ is removed from $L_v$ by moving the last entry in $L_v$ to take its place (provided the entry for $w$ is not last on the list) and decrementing LEN[$v$] by one. In the undirected case both entries must be deleted from their respective lists, the lists made compact (adjusting pointers where necessary), and both lengths decremented.

When calling SELECT($u$) we fail if LEN[$u$] = 0. Otherwise we call on the random element of the r-RAC and extract the number $x$ represented by the first $\lceil \log_2 \text{LEN}[u] \rceil$ bits of the random word. Repeating this operation we generate a succession of values $x$ until we find one in the range $1 \leqslant x \leqslant \text{LEN}[u]$. Then if $w$ is the $x$th entry for $L_v$ we delete the edge $(v, w)$ (or $\{v, w\}$) as described in the preceding paragraph and return $w$ as the value of SELECT.

During a single call of SELECT each call upon the random element has a probability of at least $\frac{1}{2}$ of producing an $x$ in the appropriate range. Thus in $Mn \log n$ calls of SELECT the probability that the random element must be called on more than (say) $4Mn \log n$ times is exponentially decreasing in $n$ (i.e. $O(n^{-\alpha})$ for any $\alpha$).

For the DHC and UHC algorithms we need to represent paths and cycles. For the DHC algorithm we need the following operations:

(i) adding a node to the end of a list

(ii) determining the ordinal position of a node within a list

(iii) splitting a list (to form a path and cycle or rejoin a path and cycle)

(iv) concatenating a list.

Each of these operations can be carried out in $O(\log n)$ time if balanced trees (e.g., 2–3 trees [1]) are used to represent paths and cycles. For the UHC algorithm we need (i), (iii), and (iv), and, in addition, the operation of list reversal. If "reversible 2–3 trees" [20] are used then this set of operations still requires only $O(\log n)$ steps per operation. (Reversible 2–3 trees are essentially 2–3 trees with an additional bit of information at each node indicating in which direction the associated subtree is to be traversed.) We therefore conclude:

THEOREM 12.1. $\forall \alpha\ \exists M, c\ \forall s, t \in V, N \geqslant cn \log n$ *the probability that* DHC($D_N$, $s$, $t$) *returns "success" within time bound* $Mn(\log n)^2$ *on a* r-RAC *is* $1 - O(n^{-\alpha})$, *and similarly for* UHC($G_N$, $s$, $t$).

*Remark.* Proposition 3.1 ensures that this Theorem holds for the corresponding $D_p$ and $G_p$.

For the PM algorithm we represent the partial matching $P$ as an $n$-vector $P[v]$ where $P[v] = w$ if $\{v, w\} \in P$ and $P[v] = 0$ if $v$ is not in $P$. It is clear that after each call of SELECT the updating of $P$ that must be performed in the remainder of step 3 can be done in constant time. Furthermore, since the values of ndp chosen by successive executions of step 2(b) are monotonically increasing, it suffices to remember the last such value and scan $P$ forwards from that point at each execution of 2(b). Hence the total time spent in step 2 is $O(n)$. We can therefore conclude the following Theorem as well as its analogue for $G_p$:

THEOREM 12.2. $\forall\alpha\ \exists M, c\ \forall N \geqslant cn \log n$ *the probability that* $\mathrm{PM}(G_N)$ *returns "success" within time* $Mn \log n$ *on a* r-RAC *is* $1 - O(n^{-\alpha})$.

*Implementation on a RAC*

It remains to analyse the complexity of the algorithms in the case that the random element of the RAC is replaced by a supply of random bits obtained by derandomization.

The derandomization process is implemented by searching the appropriate parts of the adjacency lists linearly. For successive segments of length $h = \lfloor 1/\hat{p} \rfloor$ of the integers we can thus count the number of edges that occur in the segment, identifying those which will generate $\lfloor \log_2 h \rfloor$ random bits according to the derandomization procedure. By applying Proposition 2.4(b) to bound the number of edges searched, it follows that with probability $1 - O(n^{-\alpha})$ the time spent in derandomization is $O(n(\log n)^2)$. (Note: the worst case for this procedure occurs at $p = \frac{1}{2}$, when only about one random bit is generated from each "successful" segment. If $p = O(n^{-\epsilon})$ for some $\epsilon > 0$ then we get $\Omega(\log n)$ bits from each such segment, and the time spent in derandomization can be taken to be $O(n \log n)$.)

By means of further applications of Propositions 3.1 and 3.2 we conclude the following, as well as analogues for $D_p$ and $G_p$:

THEOREM 12.3. *Theorem* 12.1 *holds for* RACs.

Since the complexity of derandomization dominates that of PM we get:

THEOREM 12.4. *Theorem* 12.2 *holds for* RACs *provided either that the additional restriction* $N = O(n^{2-\epsilon})$ *is added or that the time bound is increased to* $Mn(\log n)^2$.

Finally we note that the algorithm for UHC obtained by reduction via Proposition 3.4 to DHC also has a derandomized version with the same time bound. The original undirected graph is broken in two, and the connecting edges are used as a random source both for generating the directed graphs (edge by edge, on demand) and for running DHC on them. It is clear that the source can supply at constant cost random bits not only of probability $\frac{1}{2}$, as already shown, but also of probability exactly $p$, as is required.

## 13. CONCLUSION

We have shown that in the particular context of Hamiltonian circuits and matchings in random graphs rigorous probabilistic analysis is possible for a family of simple, efficient algorithms. Since the proof techniques formalize some fairly intuitive notions, we hope that they will be of wider application.

Concerning the particular problems we have addressed several lines of enquiry immediately recommend themselves:

(i) It is possible that a rich variety of other algorithms for these problems have probabilistic behaviour which is provably as good or better.

(ii) It is likely that the particular algorithms we give can be made more efficient in practice by various "heuristic" modifications. (For example, some preliminary experiments suggest that it might be better not to delete edges as they are explored.)

(iii) Because of our imprecision about constant factors, it is possible that the given algorithms perform much better than the analysis would suggest. There remains considerable scope for clarifying the relationships between the constant parameters by both analysis and experimentation.

(iv) Perhaps (in the spirit of Koršunov's work on UHCs [19]) a more complicated algorithm could be used to identify more accurately the threshold of density at which Hamiltonian circuits exist almost certainly in directed graphs.

## 14. Appendix: Random Access Computers

The Random Access Computer model is intended to reconcile the notion of having operations on only *finite words* with that of expressing complexities as *asymptotic* run-times. The key device used is that of making the word size dependent on the input size $n$. We first define a schema of such models. We then take a particular instance of it that appears a realistic one, in relation to present day computers, for expressing the run-times of combinatorial algorithms of low complexity. In fact for the vast majority of published algorithms the asymptotic run-times claimed would be unchanged if programmed for this particular model. In some sense we are therefore merely making concrete intuitions that already pervade the literature. A related model has, indeed, been treated explicitly by Slisenko [26].

A $\mathrm{RAC}_I(\lambda)$ is an abstract machine with *word size* $\lambda(n)$ and fixed *instruction set* $I$, where $n$ denotes the *input size* (i.e. the number of words occupied by the input) for the computation under consideration. (N.B. Each element of $I$ can be regarded as a schema specifying the instruction for each possible value of $\lambda(n)$.) A RAC is controlled by a *fixed program* (independent of $n$) consisting of a finite sequence of instructions from the set $I$. The *store* consists of $M = 2^{\lambda(n)}$ *words*, numbered from 0 to $M - 1$, each containing a string of $\lambda(n)$ binary bits (i.e. a string from $\{0, 1\}^{\lambda}$). The *time-complexity* of a computation is the number of instructions executed. Here we shall deal only with the special case in which the store is initialized to zero. [N.B. The function $\lambda(n)$ will be abbreviated to $\lambda$].

For each $\mathrm{RAC}_I(\lambda)$ there is an associated *randomized* RAC, denoted by $\text{r-RAC}_I(\lambda)$, that is obtained by having an additional *random element* instruction that in one step generates a random $\lambda$-bit binary word.

In the paper we have standardized implicitly on the $\mathrm{RAC}_J(k \log_2 n)$, where $J$ is the set described in Table 1, and $k$ is some constant (e.g. $k = 3$ is sufficient.) $J$ represents a useful set of instructions, and is not intended to be minimal in any sense. By varying $k$ we can allow the machine to compute any polynomial-space computable function. Also, $k$ has to be at least one if all the input is to be in store. [N.B. By making $\lambda$ much

larger e.g. $\lambda(n) \sim n$ or $\lambda(n) \sim 2^n$ we could discuss in a unified way some alternative machine architectures that allow much more parallel processing.]

Let $Z_M = \{0, 1, \ldots, M-1\}$. In defining $J$ we regard the contents of a word according to context *either* as a string from $\{0, 1\}^\lambda$, *or* as the integer $i \in Z_M$ which the string represents in binary. Also, we denote the "current contents" of the word numbered $j$ by $\langle j \rangle$.

TABLE I

Instruction Set $J$

| Instruction | Meaning |
|---|---|
| $W_i \leftarrow m$ | $\langle i \rangle \leftarrow m \pmod M$ |
| $W_i \leftarrow W_j + W_k$ | $\langle i \rangle \leftarrow \langle j \rangle + \langle k \rangle \pmod M$ |
| $W_i \leftarrow W_j - W_k$ | $\langle i \rangle \leftarrow \langle j \rangle - \langle k \rangle \pmod M$ |
| FETCH $(W_i, W_j)$ | $\langle i \rangle \leftarrow \langle\langle j \rangle\rangle$ |
| STORE $(W_i, W_j)$ | $\langle\langle j \rangle\rangle \leftarrow \langle i \rangle$ |
| JUMP TO $m$ IF $W_i = 0$ | Transfer control to $m$th instruction if $\langle i \rangle = 0$ |
| READ $(W_i)$ | For $m = 1, \ldots, n$: $\langle\langle i \rangle - 1 + m \rangle \leftarrow$ next input word |
| WRITE $(W_i)$ | output $\langle i \rangle$ |
| $W_i \leftarrow \neg W_j$ | $\langle i \rangle \leftarrow$ bitwise negation of $\langle j \rangle$ |
| $W_i \leftarrow W_j \wedge W_k$ | $\langle i \rangle \leftarrow$ bitwise conjunction of $\langle j \rangle$ and $\langle k \rangle$ |
| $W_i \leftarrow W_j * W_k$ | $\langle i \rangle \leftarrow \langle j \rangle * \langle k \rangle \pmod M$ |
| $W_i \leftarrow W_j / W_k$ | $\langle i \rangle \leftarrow \max\{q \in Z_M \mid q * \langle k \rangle \leqslant \langle j \rangle\}$ |
| HALT | |

Note that since the READ instruction reads all the input in a single step we can assume that the input is effectively resident in memory.

The following three propositions relate RACs to log-cost RAMs [1, 3] and Turing machines.

PROPOSITION 14.1. *For any $k$ a program of complexity $T(n)$ on a* RAC$_J(k \log_2 n)$ *can be simulated by a log-cost* RAM *program of complexity* $O(T(n)(\log n)^2 + n \log n)$.

*Proof.* In the simulation the RAM first reads the entire input sequence (assumed to have a special terminator) and deduces the value of $n$, $\lambda = \lceil k \log n \rceil$, and $M = 2^\lambda$. All this takes $O(n \log n)$ time.

In the remainder of the simulation the store of the RAC is represented by an array $A[i]$ where $i = 0, 1, \ldots, M-1$, such that for each $i$ $A[i]$ will contain $\langle i \rangle$. It remains to show that each instruction in $J$ (other than READ) can be simulated with $O((\log n)^2)$ cost.

To implement instructions we represent data either as numbers stored in a single

word, or alternatively as bit strings stored in a list of $\lambda$ consecutive words, with one bit in each. Conversion between the two can be done in $O((\log n)^2)$ time in both directions, as can be easily verified. This allows direct $O((\log n)^2)$ implementations of all the instructions. For example, for $W_i \leftarrow W_j * W_k$ we use the school algorithm, treating the multiplicand as a number and the multiplier as a bit-string. We shift the multiplicand by doubling, $O(\log n)$ times, and accumulate these numbers according to the bits of the multiplier. The final result is trimmed (mod $M$) by comparisons with a table of the first $2\lambda$ powers of 2, and subtraction. In a similar way the school algorithm for long division suffices for $W_i \leftarrow W_j / W_k$. ▮

Note that if the instruction set of the RAC is restricted to the first eight then the $(\log n)^2$ factor in the above proposition can be replaced by $\log n$. For this the use of bit-strings is omitted entirely, numbers being brought into the correct range, if necessary, by means of comparisons and additions or subtractions.

We say that a RAM program runs in polynomial space if there is a polynomial $p(n)$ such that for all inputs of size $n$ the sum of the lengths of the binary representations of all the nonzero numbers in store is less than $p(n)$ throughout the computation.

PROPOSITION 14.2. *Suppose $P$ is a log-cost* RAM *that runs in polynomial space and time $T(n)$. Then for some $k$ it can be simulated by a* $\mathrm{RAC}_J(\lceil k \log_2 n \rceil)$ *in time $O(T(n))$.*

*Proof.* To deal with the variable-length registers of a RAM we employ a simple storage allocation scheme. At any time it maintains a linked list of free *blocks* (of say two words each) and a pointer to the beginning of "unused" memory. An allocation request causes it to remove (and return) the first block from the free list, or if that is empty, to create (and return) a new block from the beginning of unused memory. A de-allocation request presents a block which is simply threaded into the free list.

The usual methods of linked list representation then allow us to represent the contents of a RAM register as a doubly-linked list of RAC words, one for the sign of the number and one for each bit of the base two representation of the absolute value of the number. When an assignment is made to a variable the list representing its old value is de-allocated block by block, and a new list allocated.

In order to address words that already have values, a tree representing lexicographically their binary RAM addresses is used. The leaves of the tree point to the lists representing their values.

It is easy to see that with these structures each RAM instruction can be simulated in time proportional to its cost on a log-cost RAM assuming appropriate input-output conventions. ▮

Finally we observe the following:

PROPOSITION 14.3. *If $P$ is a deterministic multi-tape Turing machine with time complexity bounded by $T(n) \geqslant n \log n$, where $T(n)$ is some polynomial in $n$, then for some $k$ it can be simulated by some* $\mathrm{RAC}_J(k \log n)$ *in time $O(T(n)/\log T(n))$.*

*Proof.* The analogous simulation in Theorem 5 in [14] is valid here also. ▮

## REFERENCES


1. A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, Mass. 1974.
2. S. A. COOK, The complexity of theorem proving procedures, *in* "Proceedings of the Third Annual ACM Symposium on the Theory of Computing, 1971, pp. 151–158.
3. S. A. COOK AND R. A. RECKHOW, Time-bounded random access machines, *J. Comput. System Sci.* **7** (1973), 354–375.
4. G. A. DIRAC, Some theorems on abstract graphs, *Proc. London Math. Soc.* **2** (1952), 69–81.
5. J. EDMONDS, Paths, trees, and flowers, *Canad. J. Math.* **17** (1965), 449–467.
6. P. ERDÖS AND A. RÉNYI, On random graphs I, *Publ. Math. Debrecen* **6** (1959), 290–297.
7. P. ERDÖS AND A. RÉNYI, On the existence of a factor of degree one of connected random graphs, *Acta Math. Acad. Sci. Hungar.* **17** (1966), 359–379.
8. P. ERDÖS AND J. SPENCER, "Probabilistic Methods in Combinatorics," Academic Press, New York, 1974.
9. S. EVEN AND O. KARIV, An $O(n^{2.5})$ algorithm for maximum matching in general graphs, *in* "Proceedings of the 16th IEEE Symposium on Foundations of Computer Science, 1975," pp. 100–112.
10. W. FELLER, "An Introduction to Probability Theory and its Applications," 3rd ed., Vol. I, Wiley, New York, 1968.
11. G. R. GRIMMETT AND C. J. H. MCDIARMID, On colouring random graphs, *Proc. Cambridge Philos. Soc.* **77** (1975), 313–324.
12. E. H. GUIMADY, N. I. GLEBOV, AND V. A. PEREPELICA, Algorithms with bounds for problems of discrete optimizations, *Problemy Kibernet.* **31** (1976), 35–52.
13. E. H. GUIMADY AND V. A. PEREPELICA, On statistically efficient algorithms for some extremal problems from graph theory, *in* "Tezisy III Vsesoyznoi Konferencii po Problemam Teoreticheskoi Kibernetiki, Novosibirsk, 1974," pp. 125–127.
14. J. E. HOPCROFT, W. J. PAUL, AND L. G. VALIANT, On time versus space and related problems, *in* "16th IEEE Symposium on the Foundations of Computer Science, 1975," pp. 57–64.
15. R. M. KARP, Reducibility among combinatorial problems, *in* "Complexity of Computer Computations" (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–104, Plenum, New York, 1972.
16. R. M. KARP, The probabilistic analysis of some combinatorial search algorithms, *in* "Algorithms and Complexity" (J. F. Traub, Ed.), Academic Press, New York, 1976.
17. J. KOMLÓS AND E. SZEMERÉDI, private communication.
18. A. D. KORŠUNOV, On the power of some classes of graphs, *Soviet Math. Dokl.* **11** (1970), 1100–1104.
19. A. D. KORŠUNOV, Solution of a problem of Erdös and Rényi on Hamiltonian cycles in non-oriented graphs, *Soviet Math. Dokl.* **17** (1976), 760–764.
20. G. LEV, Reversible concatenable lists, manuscript, Department of Computer Science, University of Edinburgh, 1976.
21. L. A. LEVIN, Universal combinatorial problems, *Problemy Peredači Informacii* **9** (1972), 115–116.
22. V. A. PEREPELICA, On two problems from the theory of graphs, *Soviet Math. Dokl.* **11** (1970), 1376–1379.
23. L. POSA, Hamiltonian circuits in random graphs, *Discrete Math.* **14** (1976), 359–364.
24. M. O. RABIN, Probabilistic algorithms, *in* "Algorithms and Complexity" (J. F. Traub, Ed.), Academic Press, New York, 1976.
25. C. P. SCHNORR, Optimal algorithms for self-reducible problems, *in* "Proceedings of the Third International Colloquium on Automata, Languages and Programming" (S. Michaelson and R. Milner, Eds.), pp. 322–337, Edinburgh Univ. Press, Edinburgh, 1976.
26. A. O. SLISENKO, String-matching in real time, Preprint R-7-77 of LOMI, USSR Academy of Science, Leningrad, 1977.

27. R. SOLOVAY AND V. STRASSEN, A fast Monte-Carlo test for primality, *SIAM J. Comput.* **6** (1977), 84–85.
28. E. M. WRIGHT, For how many edges is a digraph almost certainly Hamiltonian? *Proc. Amer. Math. Soc.* **41**, 2 (1973), 384–388.
29. E. M. WRIGHT, Graphs on unlabelled nodes with a given number of edges, *Acta Math.* **126** (1971), 1–9.