

On a Simple Randomized Algorithm for Finding a 2-Factor in Sparse Graphs

Gopal Pandurangan *

Abstract

We analyze the performance of a simple randomized algorithm for finding 2-factors in directed Hamiltonian graphs of out-degree at most two and in undirected Hamiltonian graphs of degree at most three. For the directed case, the algorithm finds a 2-factor in $O(n^2)$ expected time. The analysis of our algorithm is based on random walks on the line and interestingly resembles the analysis of a randomized algorithm for the 2-SAT problem given by Papadimitriou [15]. For the undirected case, the algorithm finds a 2-factor in $O(n^3)$ expected time. We also analyze *random* versions of these graphs and show that cycles of length $\Omega(n/\log n)$ can be found with high probability in polynomial time. This partially answers an open question of Broder *et al.* [4] on finding hidden Hamiltonian cycles in sparse random graphs and improves on a result of Karger *et al.* [13].

Keywords: Graph Algorithms, Randomized Algorithms, Random Walks, Sparse Graphs, 2-Factor, Hamiltonian Cycle, Long Cycle.

1 Introduction

In this paper, we analyze the performance of a simple randomized algorithm in directed Hamiltonian graphs of out-degree at most two and in undirected Hamiltonian graphs of degree at most three. For the directed case, the algorithm finds a Hamiltonian cycle or terminates with some 2-factor in $O(n^2)$ expected time (cf. Theorem 1). In fact, it finds a 2-factor in any directed graph with out-degree at most 2 that has a 2-factor in $O(n^2)$ expected time. The analysis of our randomized algorithm is based on analyzing the expected time of a random walk on a line to hit one end of the barrier (also called the “gambler’s ruin” problem, see e.g., [12]). Papadimitriou [15] used a similar random walk based approach to analyze a simple randomized algorithm for the 2-SAT problem. The randomized algorithm was surprisingly simple and yielded a new $O(n^2)$ expected time algorithm for 2-SAT.¹ This raised a curious question whether a similar approach can be used for other problems. For the undirected case, the algorithm terminates with a 2-factor in $O(n^3)$ expected time, (cf. Theorem 2). The proof interestingly involves studying a random walk on a graph with

*Department of Computer Science, Purdue University, 250 N. Univ. St., West Lafayette, IN 47907, USA. Email: gopal@cs.purdue.edu.

¹The randomized approach for 2-SAT by Papadimitriou was independently used by Feder [6] to give an $O(m^2)$ expected time algorithm for stable marriage and its generalizations.

changing edges — thus “standard” results on random walks don’t directly apply, however, by appealing to special structure of the walk (in particular, its reversibility) we are able to show the desired bound. One may consider using a “gambler’s ruin” type of random walk to “discover” the Hamiltonian edges (as in proof of Theorem 1); unfortunately, this yields only an exponential time algorithm to find a 2-factor.² Although somewhat more efficient deterministic algorithms exist for finding 2-factors in graphs (e.g., an $O(\sqrt{n}|E|)$ time algorithm [11]), our randomized algorithm is extremely simple and naturally *reuses* already found paths (i.e., makes only local changes in each step). Reusing already found paths is intuitive and has been a feature of the rotation based algorithms for the $G_{n,p}$ random graphs (see e.g., [1]).

Our randomized algorithm finds application in finding long cycles or paths in certain sparse *random* graphs used to model NMR data of proteins [5]. In fact, one of our motivations for this work (see [5]) is the problem of finding Hamiltonian or long cycles/paths in sparse Hamiltonian graphs. Finding Hamiltonian cycles in graphs is a difficult NP-hard problem and remains NP-hard even when restricted to sparse Hamiltonian graphs, e.g., directed Hamiltonian graphs with maximum out-degree two [16] or undirected Hamiltonian graphs with degree at most three [10]. The problem has been shown hard to approximate in directed graphs: Bjorklund *et al.* [3] show that it is not possible to find paths or cycles of superpolylogarithmic length even in constant out-degree Hamiltonian graphs unless Satisfiability can be solved in subexponential time. For undirected Hamiltonian graphs, for a long time, the best known results yielded only paths of polylogarithmic length (e.g., [2]), even when restricted to bounded-degree graphs. Gabow [9], recently, gave a polynomial algorithm that finds paths and cycles of superpolylogarithmic length: he showed that if a graph has a cycle of length ℓ , then one can find in polynomial time a cycle of length $\exp(\Omega(\sqrt{\log \ell / \log \log \ell}))$. This is the best bound known at the present time, and is improved for graphs of degree bounded by d to $\ell^{\Omega(1/\log d)}$ by Feder and Motwani [8]. In particular, they show that a cycle of length $n^{\Omega(1/\log \log n)}$ can be found in Hamiltonian graphs in polynomial time. For the special case of undirected Hamiltonian graphs with degree at most *three* Feder *et al.* [7] give a polynomial time algorithm to find a cycle of length at least $n^{(\log_3 2)/2}$.

While our randomized algorithm is guaranteed to find only a 2-factor (even in a Hamiltonian graph), it turns out that in sparse *random* versions of these graphs, i.e., directed 2-regular random Hamiltonian graphs and undirected 3-regular random Hamiltonian graphs, the algorithm can find a long cycle. We show that the algorithm finds a cycle of length $\Omega(n/\log n)$ with high probability (w.h.p.)³ in polynomial time (cf. Theorem 3). This is because we show that (cf. Lemma 1) finding a 2-factor is enough to find a cycle of length at least $\Omega(n/\log n)$ in these random graphs. Our results partially answer an open question of Broder *et al.* [4] who showed that for *random* Hamiltonian graphs with average degree much larger than three, there is a polynomial time algorithm that actually finds an Hamiltonian cycle. However, they left open the question of whether an Hamiltonian path/cycle or even a long path/cycle can be found in sparse random graphs, e.g., graphs of degree 3. Our result on 3-regular random Hamiltonian graphs improves on a result of Karger *et al.* [13] who give a polynomial algorithm which finds a path of length only $\Omega(\sqrt{n}/\log n)$ with high probability. It will be interesting to give a polynomial algorithm that will actually find an Hamiltonian

²For undirected Hamiltonian graphs of degree at most three, this can be shown to take $O(2^{n/2})$ expected time.

³with probability at least $1 - 1/n^{\Omega(1)}$.

cycle in the above types of random graphs.

2 The Randomized Algorithm and its Analysis

For a graph $G = (V, E)$, define a *fragment cover* C to be a set of simple paths or cycles (each path or cycle is called a *fragment* and can be of length zero) such that each vertex $v \in V$ appears in exactly one fragment. We say that an edge belongs to C if it appears in any of the fragments in C . A fragment cover will be an Hamiltonian cycle (or a 2-factor) if there is only one fragment which forms a cycle (or if fragments form disjoint cycles). We discuss a simple randomized algorithm for directed graphs; later we will modify the algorithm to handle undirected graphs. Refer Figure 1. Let C denote a fragment cover. Initially $C = V$, i.e., each vertex is a fragment by itself. Let $\text{succ}(u, C)$ and $\text{pred}(u, C)$ denote the successor and predecessor of u in a cover C (these can be *null*). In every step of the while loop, we extend a vertex (say u) which does not have a successor (i.e., an outgoing edge), by choosing an outgoing edge with uniform probability (i.e., with probability $1/d(u)$ if $d(u)$ is the out-degree of u), and potentially swapping a chosen edge for a previously chosen edge. In the algorithm, the set $V - W$ maintains the vertices with no successors. We repeat the while loop till every vertex has a successor. Thus when the algorithm terminates C will consist of either a Hamiltonian cycle or a collection of vertex-disjoint cycles covering V (i.e., every vertex will belong to a non-trivial cycle) — a 2-factor of G .

```
Given  $G = (V, E)$ 
Let initial fragment cover  $C = V$ 
Let  $W = \emptyset$ 

1 While  $V - W$  is not empty do
2   Choose  $u$  from  $V - W$ 
3   Choose an edge  $(u, v)$  uniformly at random among all incident edges
4   If predecessor of  $v$  in  $C$  is null then
5     Join the two fragments in  $C$ 
6   Else
7     Create two fragments in  $C$ :  $\langle \dots u, v \dots \rangle$ , and  $\langle \dots, \text{pred}(v) \rangle$ 
8     Remove  $\text{pred}(v)$  from  $W$ 
9   Add  $u$  to  $W$ 
10 endwhile
```

Figure 1: The randomized algorithm for directed graphs.

We analyze the algorithm on sparse directed graphs which have a 2-factor, in particular those with out-degree bound of two. (As mentioned before, it is NP-hard to find a Hamiltonian cycle if the graph contains a Hamiltonian cycle.)

Theorem 1 *Let G be a directed graph on n vertices such that each vertex has at most two outgoing edges and let G contain a 2-factor (e.g., a Hamiltonian cycle). Then the randomized algorithm will terminate with a 2-factor of G in expected $O(n^2)$ steps or in $O(n^2 \log n)$ steps w.h.p.*

Proof: Let H be a 2-factor in G . We focus on the algorithm finding H . We view the algorithm as a random walk on a line numbered $0, \dots, n$ as follows. In some step (one execution of the while loop), say t , the algorithm is stationed at number k where k is the number of “correct” edges, i.e., those belonging to H ; in the beginning $k = 0$. We claim that in step $t + 1$, the algorithm can move to $k + 1$ with probability at least $1/2$. The reasoning is as follows. Let u be the vertex (does not have a successor yet) considered this step, then the probability that the correct edge, say, (u, v) is chosen is at least $1/2$. Now, does this affect the probability of some other edge? The only other edge which is affected is $(\text{pred}(v, C), v)$. There are two cases: (a) $\text{pred}(v, C)$ is null, in which case we have $k + 1$ correct edges; (b) otherwise, we lose the edge $(\text{pred}(v, C), v)$; but the probability that that edge is correct *conditioned* on the fact that (u, v) is correct is 0. Thus with probability at least $1/2$ we add one correct edge. With probability at most $1/2$, the algorithm can move to $k - 1$ or stay at k . We are interested in the expected number of steps needed to reach n . By setting up a difference equation (for example, see [12, pp. 73–74]), we can show that the expected number of steps needed to reach n is at most n^2 . The high probability result follows from repeating the algorithm if it does not terminate in $2n^2$ steps.

The above analysis assumes that in every step there is a vertex that does not have a successor. If not, this implies that C will consist of a set of disjoint (non-trivial) cycles covering every vertex, i.e., a 2-factor. \square

A few simple modifications of the randomized algorithm of Figure 1 are needed to apply it for undirected graphs. See Figure 2. C is a fragment cover as before and initially C consists of $|V|$ zero length fragments. Let $r(v)$ denote the degree of v restricted to the edges in C . Note that $r(v) \leq 2$. We start with an arbitrary vertex s . In each step, we try to extend the end vertex of P (call it u), by choosing an edge uniformly at random among the incident edges (if $r(u) = 1$, we don’t choose the edge that is already in C). Let the chosen edge be (u, v) . There are three possibilities. If $r(v) = 0$ or $r(v) = 1$ we extend u to v (step 8 or 10); if $r(v) = 1$, this will complete a cycle at s and we start extending from a new vertex with r -value 0, if one exists. If $r(v) = 2$, then there are two possible scenarios: v is in P or v is in a simple cycle. In either case, we randomly delete one of the incident edges of v and add (u, v) to C (step 14). Note that if v is in P , this can create a new cycle or a path which is a *rotation* (see e.g., [1]) of P and if v is in a cycle it creates a (longer) path starting from s .

We now analyze the algorithm on *Hamiltonian* graphs with degree at most 3 and show that the randomized algorithm will find a 2-factor in polynomial ($O(n^3)$) expected time. (However, as noted before, finding an Hamiltonian cycle is NP-hard.)

Theorem 2 *Let $G = (V, E)$ be an undirected Hamiltonian graph on n vertices where each vertex has degree at most three. Then the randomized algorithm of Figure 2 will terminate with a 2-factor of G in expected $O(n^3)$ steps or in $O(n^3 \log n)$ steps w.h.p.*

```

Given  $G = (V, E)$ 
Let initial cover  $C = V$ 
 $r$ -value of  $v$  denoted by  $r(v) (\leq 2)$  denotes the degree of a vertex  $v$  restricted to  $C$ 
Fix an arbitrary starting vertex  $s$ 

1 While there is a vertex with  $r$ -value less than 2 do
2     Let  $u$  be the end point of the path starting from  $s$  (can be  $s$  itself)
3     If  $r(u) = 0$  then
4         Choose an edge  $(u, v)$  uniformly at random among all incident edges
5     Else (i.e.,  $r(u) = 1$  and let  $(u, w)$  appear in a fragment in  $C$ )
6         Choose an edge  $(u, v)$  uniformly at random among all incident edges except  $(u, w)$ 
7     If  $r(v) = 0$  then
8         Extend  $u$  to  $v$ 
9     If  $r(v) = 1$  then
10        Extend  $u$  to  $v (= s)$  and complete the cycle
11        If  $C$  has  $n$  edges then stop
12        Else set  $s$  to be an arbitrary vertex with  $r$ -value 0
13    If  $r(v) = 2$ 
14        Delete one of the two incident edges of  $v$  with equal probability and add  $(u, v)$  to  $C$ 
15 endwhile

```

Figure 2: The basic randomized algorithm for undirected graphs.

Proof: It is clear that the algorithm will terminate with a 2-factor since every vertex in C will have a degree of two. We bound the expected time needed for termination. For the analysis, we divide the algorithm into phases as follows. In a phase, we always try to extend a path starting from a fixed vertex, say u , till u becomes part of a cycle. In the next phase, we pick an arbitrary vertex (say w) such that $r(w) = 0$ (if no such vertex exists then the algorithm is finished — step 11) and try to extend a path starting at w and so on.

In some step (i.e., one execution of the while loop) of the randomized algorithm of Figure 2, let $k < n - 1$ be the number of edges in the fragment cover C . We note that the number of edges in C in any subsequent step can never decrease: we either add a new edge to C (in the case when $r(v) = 0$) or keep the same number of edges (in the case when $r(v) = 2$ we add and delete an edge). What is the expected number of steps before we increase the number of edges in C to $k + 1$? Let C' be the subgraph of C induced by the vertices with r -value at least 1. Let the algorithm be currently stationed at vertex x and let the start vertex at the beginning of this phase be s (s could be x itself). Note that if we reach (i.e., make an end vertex) any vertex which has a neighbor in $V - C'$ (we will call such vertices as *bridge* vertices), then with probability at least $1/2$ we will add one more edge to C . There are at least *two* bridge vertices in C' since G is Hamiltonian and we have less than n vertices in C' . It is clear that the algorithm will eventually visit some bridge vertex in C' (starting from x) unless we reach and complete a cycle with s before reaching y . This is easy to see, because, if we just keep extending a path starting from s , since G is Hamiltonian, we will eventually visit some vertex in $V - C'$ (unless of course, we complete the cycle at s and end this phase).

Assume that in the current phase the algorithm is trying to extend a path (call it P) starting from some fixed vertex s and, without loss of generality, let $r(s) = 1$. Let x be the current endpoint of the path starting from s . In each step of the algorithm in this phase, the *end vertex of P* changes as follows: if v_1 is the current end vertex, there is a transition from v_1 to v_2 (i.e., we can “reach” v_2) if there is a $v_3 \in C'$ such that $(v_1, v_3) \in E(G)$ and $(v_3, v_2) \in E(C')$ — this captures how the end vertex of P can change in step 14. (Note that, if there is a transition from (v_1, v_2) then there can also be a transition from (v_2, v_1) , and hence the walk is reversible. However, note that the edges of C' are changing as the random walk proceeds, thus we cannot directly apply the standard results from random walks on graphs.)

We show by induction on l — the number of vertices in C' (we also include the start vertex in C' , even though its r -value may become zero during the phase) — that we will add one more edge in at most l^2 expected steps (either by adding a new vertex to C' or by completing the cycle to the start vertex s). The induction hypothesis consists of two assumptions: (1) starting from any vertex x (as end vertex), the algorithm will add a new edge in at most l^2 expected steps on sets C' of size $\leq l$ and (2) starting from any bridge vertex u , the algorithm will add a new edge in at most l expected steps on sets C' of size $\leq l$.

The base case ($l = 1$) is trivial. We will show that the hypothesis is true for sets of size $l + 1$. Let x be the current end vertex and let $|C'| = l + 1$. We know that, since G is Hamiltonian, there are at least two bridge vertices in C' (say v_1 and v_2) and at least one of them (say v_1) is reachable from x (note that v_1 can be x itself, and assume that $v_2 \neq x$). If only one of them is reachable, then we remove the unreachable vertex and merge the edges (in C') incident on it and apply the induction hypothesis. Otherwise, we proceed by assuming that both v_1 and v_2 are reachable. For the sake of argument pretend that the random walk is made by the algorithm on a graph H obtained from C' by removing v_2 and merging its two⁴ incident edges (in C') into a new edge — call it f . Using part (1) of the induction hypothesis on H , the algorithm will add a new edge in expected l^2 steps unless it uses (removes) edge f before that. That is, we reach v_2 ⁵ and with probability at least $1/2$ we add a new edge. Otherwise, consider the random walk on the graph obtained by merging v_1 and its incident edges (in C'). Now starting from v_2 and using part (2) of the hypothesis, we will either add a new edge in l steps or reach v_1 ⁶; if we reach v_1 , we repeat the argument with v_1 as the starting bridge vertex. Thus, overall, the expected number of steps needed to add an edge starting from a bridge vertex is at most $(1/2)l + (1/2)(2l + 1) = l + 1$; thus induction step for part (2) is proved.⁷ To show for part (1), we see that the expected number of steps needed to add an edge starting from any vertex is at most $l^2 + l + 1 \leq (l + 1)^2$.

If $l = n$ (i.e., the number of edges in C' is $k = n - 1$ and the algorithm will be in the last phase), then we only need to close the cycle with the start vertex of this phase. In this case, we compute the expected time to reach a neighbor of s . The above argument does not directly extend since there are no bridge vertices.

⁴If v_2 is s , then there will be one incident edge in C' .

⁵Note this is the only way to remove f , as v_2 has degree three and two of its neighbors are in C' and the remaining neighbor is in $V - C'$ (thus the only other way to remove f is by reaching this “outside vertex”, in which case we would have already added an edge). However, if the degree is greater than three, this argument will not work.

⁶Note that the reversibility of the walk guarantees that v_1 and v_2 are mutually reachable if they are both reachable from x .

⁷We remark that, if the degree is greater than three, a similar induction argument gives only an exponential time bound.

However, there are at least two vertices that we can use to complete the cycle with s : the two neighbors of s in the Hamiltonian cycle of G — these two vertices serve the same role as v_1 and v_2 in the above argument as follows. Again, let the end vertex at the beginning of this phase be x . We will assume, for now, that $r(s)$ never becomes zero in this phase. Let v ($\neq v_1$, say) be the vertex adjacent to s in the path P . If one of v_1 or v_2 is unreachable from x , then we merge the two incident edges on that vertex and use induction hypothesis. Hence we will assume that all vertices are reachable. As before, we merge the two incident edges on v_2 (into a single edge f) and argue that within $l = (n - 1)^2$ steps we either close a cycle with s or we reach s or we reach v_2 before we remove edge f . This is because there are only two ways of removing f , either reaching v_2 (in which case with probability at least $1/2$ we close the cycle) or reaching s and adding the edge (s, v_2) . In the former case, it follows that in n^2 steps we will either reach s or complete a cycle with s . On the other hand, if we reach s , then $r(s) = 0$ which implies that all the rest of vertices are part of some cycle. An easy fix to include s also in a 2-factor is to add a dummy start vertex s' by subdividing an Hamiltonian edge containing s which is not in P . That is, if (s, v_2) is an Hamiltonian edge not in P at the beginning of the (last) phase, then we remove the edge (s, v_2) and add the edges (s', s) and (s', v_2) to G , with s' being the new start vertex (and (s', s) is added to P). Since we don't know which of the (possibly) three incident edges of s belongs to the Hamiltonian cycle, we try all three possibilities in parallel.

Summing up the number of steps needed to add each edge, C will have n edges in expected $\sum_{l=1}^n l^2 = O(n^3)$ steps. \square

Remark: Unlike Theorem 1, it is not clear whether the above Theorem applies not just for Hamiltonian graphs, but for any graph (with out-degree at most 3) which contains a 2-factor. But the above proof technique works only for Hamiltonian graphs: here the existence of bridge vertices is guaranteed till the penultimate phase. However, it will be interesting to show that the algorithm (or some slight modification of it) only needs the presence of a 2-factor in G to find a 2-factor in polynomial time.

3 Random Hamiltonian Graphs

A directed 2-regular random Hamiltonian graph can be viewed as a directed Hamiltonian cycle plus a random permutation. An undirected 3-regular random Hamiltonian graph can be viewed as a Hamiltonian cycle with a random perfect matching added. We refer to the edges on the Hamiltonian cycle as *Hamiltonian edges*, and refer to the other edges as *random edges*. We show that finding a 2-factor is enough to find a cycle of length at least $\Omega(n/\log n)$ in these random graphs. Thus our randomized algorithm will find a long cycle in these graphs in polynomial time.

Lemma 1 *A directed 2-regular random Hamiltonian graph or an undirected 3-regular random Hamiltonian graph of n vertices can be partitioned into at most $O(\log n)$ vertex-disjoint cycles w.h.p. Thus any 2-factor will consist of a cycle of length $\Omega(n/\log n)$ w.h.p. (Here, w.h.p. is with respect to the probability space of all random graphs of each type.)*

Proof: Consider a 2-factor in a directed 2-regular random Hamiltonian graph. Let the 2-factor consist of b random edges ($b \geq 0$) and $n - b$ Hamiltonian edges. We show that the number of vertex-disjoint cycles in a 2-factor is $O(\log n)$ w.h.p. This is trivially true when $b = 0$ (the 2-factor is simply the Hamiltonian cycle). If $b = n$, then the 2-factor is the graph of a random permutation of n numbers and thus the number of disjoint cycles is $O(\log n)$ in expectation and w.h.p. We claim that this is still true if we add any Hamiltonian edges for the following reason. Pick an arbitrary set of $n - b$ Hamiltonian edges and consider the $b (> 0)$ random edges emanating from the b vertices that do not have successors — call this collection of b vertices as V' . These Hamiltonian and random edges will form a 2-factor only if the b random edges terminate in V' as well. Conditioned on this, note that each vertex in V' is equally likely to be the endpoint for a random edge. Thus, the number of vertex-disjoint cycles is number of disjoint cycles in a random permutation with b nodes and the b random edges. Hence, the expected number of cycles is $O(\log b)$ and a Chernoff bound [14][page 68] shows that the number of cycles is at most $c \log n$ with probability at least $1 - 1/n^3$, for a suitably large constant c . The above argument shows that if we pick a 2-factor then the number of vertex-disjoint cycles is $O(\log n)$ w.h.p. But there can be many *different* 2-factors possible and there are dependencies between them. To handle this, we bound the number of different 2-factors and show that w.h.p. all 2-factors will have at most $O(\log n)$ vertex-disjoint cycles.

There can be at most $O(n^2)$ (i.e., polynomial in n) different 2-factors in a directed 2-regular random Hamiltonian graph w.h.p. This is because the expected number of different 2-factors is at most $\sum_{b=1}^n \binom{n}{b} b! (1/(n(n-1) \dots (n-b+1))) = n$, since b random edges can be inserted in place of b Hamiltonian edges chosen in $\binom{n}{b}$ ways and there can be at most $b!$ different 2-factors each occurring with probability at most $1/(n(n-1) \dots (n-b+1))$. By Markov's inequality, with probability at least $1 - 1/n$ there are at most n^2 different 2-factors. Now, let (“bad”) event A denote that some 2-factor in G has more than $\Theta(\log n)$ vertex-disjoint cycles and event B denote that there are at most $O(n^2)$ different 2-factors. To bound $\Pr(A)$ we use $\Pr(A) \leq \Pr(A|B) + \Pr(\bar{B})$. $\Pr(\bar{B}) \leq 1/n$ and $\Pr(A|B) \leq O(1/n)$ by applying the union bound, since with probability at most $1/n^3$, a 2-factor has more than $\Theta(\log n)$ disjoint cycles and we condition that there are at most $O(n^2)$ different 2-factors.

A similar argument works for a undirected 3-regular random Hamiltonian graph. □

If we apply our randomized algorithm it will produce some 2-factor. But Lemma 1 shows that w.h.p. all 2-factors will consist of a cycle of length $\Omega(n/\log n)$. Thus we have the following theorem.

Theorem 3 *Given a directed 2-regular random Hamiltonian graph, the randomized algorithm of Figure 1 will find a cycle of length $\Omega(n/\log n)$ in $O(n^2 \log n)$ time w.h.p. (with respect to both cycle length and time — note that the probability space is determined both by the space of all such random graphs and the random choices made by the algorithm). Given a undirected 3-regular random Hamiltonian graph, the randomized algorithm of Figure 2 will find an Hamiltonian cycle of length $\Omega(n/\log n)$ in $O(n^3 \log n)$ time w.h.p.*

Acknowledgments. We thank the anonymous referees for their comments and pointers to references. We are also grateful to one of the referees for pointing out an error in an earlier version of the paper and for suggestions for improving the presentation.

References

- [1] D. Angluin and L. Valiant. Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings, *Journal of Computer and System Sciences*, **18**, 1979, 155-193.
- [2] A. Bjorklund and T. Husfeldt. Finding a Path of Superlogarithmic Length, *SIAM J. on Comput.*, **32**(6), 2003, 1395-1402.
- [3] A. Bjorklund, T. Husfeldt, and S. Khanna. Approximating Longest Directed Path, *Electronic Colloquium on Computational Complexity*, Report No. 32, 2003.
- [4] A. Broder, A. Frieze, and E. Shamir. Finding Hidden Hamilton Cycles, *Random Structures and Algorithms*, **5**, 1994, 395-410.
- [5] C. Bailey-Kellog, S. Chainraj, and G. Pandurangan. A Random Graph Approach for NMR Sequential Assignment, in *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, 2004.
- [6] T. Feder. Stable Networks and Product Graphs, *Memoirs Amer. Math. Society* **116** (555), 1995.
- [7] T. Feder, R. Motwani, and C. Subi. Approximating the Longest Cycle Problem in Sparse Graphs, *SIAM J. Comput.*, **31**, 2002, 1596-1607.
- [8] T. Feder and R. Motwani. Finding Large Cycles in Hamiltonian Graphs, to appear in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.
- [9] H.N. Gabow. Finding Paths and Cycles of Superpolylogarithmic Length, In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC)*, 2004, 407-416.
- [10] M.R. Garey, D.S. Johnson, and R.E. Tarjan. The Planar Hamiltonian Circuit Problem is NP-complete, *SIAM J. Comput.*, **5**, 1976, 704-714.
- [11] A. Gibbons. *Algorithmic Graph Theory*, Cambridge University Press, 1985.
- [12] G. Grimmett and D. Stirzaker. *Probability and Random Processes*, second edition, Oxford University Press, 1992.
- [13] D. Karger, R. Motwani, and G.D.S. Ramkumar. On Approximating the Longest Path in a Graph, *Algorithmica*, **18**, 1997, 82-98.
- [14] R. Motwani and P. Raghavan. *Randomized Algorithms*, Cambridge University Press, 1995.
- [15] C.H. Papadimitriou. On Selecting a Satisfying Truth Assignment, In *Proceedings of the 32nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, 1991, 163-169.
- [16] J. Plesnik. The NP-completeness of the Hamiltonian cycle Problem in Planar Digraphs with Degree Bound Two. *Information Processing Letters*, **8**(4), 1979, 199-201.