Finding a Hamilton cycle fast on average using rotations and extensions

Yahav Alon * Michael Krivelevich †

October 23, 2019

Abstract

We present an algorithm *CRE*, which either finds a Hamilton cycle in a graph G or determines that there is no such cycle in the graph. The algorithm's expected running time over input distribution $G \sim G(n,p)$ is $(1+o(1))n/p$, the optimal possible expected time, for $p = p(n) \geq 70n^{-\frac{1}{2}}$. This improves upon previous results on this problem due to Gurevich and Shelah, and to Thomason.

1 Introduction

Hamilton cycles are a central topic in modern graph theory, a fact that extends to the field of random graphs as well, with numerous and diverse results regarding the appearance of Hamilton cycles in random graphs obtained over many years.

Consider the random graph model $G(n,p)$, in which every one of the edges of K_n is added to G with probability p independently of the other edges. A classical result by Komlós and Szemerédi [12], and independently by Bollobás [3], states that a random graph $G \sim G(n,p)$, with $np - \ln n - \ln\ln n \to \infty$, is with high probability Hamiltonian. It should also be noted that if $np - \ln n - \ln\ln n \to -\infty$ then with high probability $\delta(G) \leq 1$, and thus G is not Hamiltonian.

In fact, a stronger result was proved by Bollobás in [3] and by Ajtai, Komlós and Szemerédi in [1]. It states that the *hitting time* of graph Hamiltonicity is with high probability equal to the hitting time of the property $\delta(G) \geq 2$. In other words: if one adds edges to an empty graph on n vertices in a random order, then with high probability the exact edge whose addition to the graph has increased its minimal degree to 2, has also made the graph Hamiltonian.

*‡School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, 6997801, Israel. Email: yahavalo@mail.tau.ac.il.

†School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, 6997801, Israel. Email: krivelev@tauex.tau.ac.il. Partially supported by USA-Israel BSF grant 2014361, and by ISF grant 1261/17.

In light of this, one can ask whether there exists a computationally efficient way to find a Hamilton cycle in a graph G, or to determine that it contains none, provided that G is sampled from the probability space $G(n,p)$ with $np - \ln n - \ln\ln n \to \infty$.
The answer to this question differs greatly depending on how one defines the term "computationally efficient".
For example, if our interest lies in finding an algorithm with a fast worst case time complexity, that is, its running time on any input is bounded by some "small" function of the number of vertices n, we might get disappointed. This is due to the fact that the graph Hamiltonicity problem is a well known NP-complete problem (see e.g. [8]), and as such no polynomial time algorithm solving it is known. In fact, the best known worst case complexity algorithm is achieved by dynamic programming algorithms (see Bellman [2] and Held, Karp [10]), with asymptotic time $O\left(2^n \cdot n^2\right)$.
That said, different models of complexity may yield very different results. Consider for example a model in which an algorithm is allowed to return the result *"failure"*, admitting that it has failed to find a Hamilton cycle in the input graph (without providing a proof that there is none), under the condition that if $p \geq f(n)$ and $G \sim G(n,p)$ then the probability that the algorithm fails on input G is of order $o(1)$.
In this model, much faster algorithms are available. A notable example is given in a 1987 paper by Bollobás, Fenner and Frieze [4], who present an algorithm *HAM1* with time complexity $O\left(n^{4+\varepsilon}\right)$ with $\varepsilon > 0$ arbitrarily small, that either finds a Hamilton cycle or returns *"failure"*. They further show that if the input graph G is distributed $G \sim G(n,p)$, for any $p = p(n)$, then

$$\lim_{n\to\infty} Pr[\textit{HAM1} \text{ finds a Hamilton cycle in } G] = \lim_{n\to\infty} Pr[G \text{ is Hamiltonian}].$$

Combined with the above stated fact that if $np - \ln n - \ln\ln n \to \infty$ then G is with high probability Hamiltonian, this means that for $p \geq \frac{\ln n + \ln\ln n + \omega(1)}{n}$ the probability that *HAM1* returns *"failure"* is indeed $o(1)$.
Another example of a fast algorithm that is not likely to return *"failure"* is given in [5], where the authors choose to measure the complexity by the number of positive edge query results the algorithm requires. They show an algorithm that requires $(1 + o(1))n$ successful queries, and fails with probability $o(1)$ on graphs distributed according to $G(n,p)$, with $p \geq \frac{\ln n + \ln\ln n + \omega(1)}{n}$.

An intuitive measure of complexity which seems interesting to consider is the *expected* running time. Denote by $T_A(G)$ the running time of some algorithm A on an input graph G. Say $G \sim G(n,p)$, how small can $\mathbb{E}[T_A(G)]$ be?
If we assume that there is no polynomial time algorithm that finds a Hamilton cycle in a graph, then finding an algorithm with polynomial expected running time is in some sense a more difficult problem than that of finding a polynomial time algorithm that fails with probability $o(1)$: if the expected time is polynomial, it means that those cases on which the running time is super-polynomial take up at most $n^{-\omega(1)}$ of the probability space. So such an algorithm can be used to construct a

polynomial time algorithm that returns *"failure"* with probability $n^{-\omega(1)}$.
Bollobás, Fenner and Frieze [4] used their algorithm *HAM1* to construct a slightly modified algorithm *HAM*, which applies an exponential running time algorithm on inputs on which *HAM1* returned *"failure"*, and prove that the expected running time of *HAM* on $G \sim G\left(n, \frac{1}{2}\right)$ is polynomial in n.
Gurevich and Shelah [9] improved upon this result, by presenting an algorithm *HPA*, which finds a Hamiltonian $s-t$ path in a graph G, with a linear expected running time, where this time the input is assumed to be distributed according to distribution $G(n,p)$, with $p \in [0,1]$ being a constant (not necessarily $\frac{1}{2}$). This can easily be altered into an algorithm that finds a Hamilton cycle rather than a Hamilton $s-t$ path. They did this by presenting three consecutive algorithms *HPA1, HPA2, HPA3*, such that failure of one algorithm to find a Hamilton $s-t$ path results in the next one being called, and such that *HPA1* takes linear time and

$$Pr[HPAi \text{ fails on } G] \cdot \mathbb{E}\left[T_{HPA(i+1)}(G)\right] = O(n).$$

They further show that their result is optimal for this range of p, by proving a stronger claim: If A is an algorithm for finding a Hamilton cycle and $p \geq \frac{3\ln n}{n}$, $G \sim G(n,p)$, then $\mathbb{E}\left[T_A(G)\right] \geq n/p$. This result can be obtained by observing that in order to find a Hamilton cycle in a graph G, the algorithm must sample at least n existing edges of G, which means that the expected number of queried pairs of vertices in A must be at least the expected number of queries required for finding n edges, which is exactly n/p.
Further improvement was later given by Thomason [14], who presented an algorithm A, similarly constructed of three consecutive algorithms *A1,A2,A3*. The expected running time of A is asymptotically optimal up to multiplication by a constant (that is $\mathbb{E}\left[T_A(G)\right] = O(n/p)$), for a wider class of random graphs: whenever $p \geq 12n^{-\frac{1}{3}}$.
For further reading on the algorithmic aspects of random graphs, including Hamiltonicity, we refer to [7].
In this paper we present a new algorithm *CRE* (Cycle rotation extension) for finding a Hamilton cycle, and prove that if $p \geq 70n^{-\frac{1}{2}}$ and $G \sim G(n,p)$ then $\mathbb{E}\left[T_{CRE}(G)\right] = (1+o(1))n/p$. This constitutes a substantial progress in a long-standing open problem on Hamiltonicity of random graphs (see e.g., **Problem 16** in [6]).
Formally, we prove the following main result:

Theorem 1. *Let $p \geq 70n^{-\frac{1}{2}}$ and let $G \sim G(n,p)$. There is an algorithm for finding a Hamilton cycle in a graph, with expected running time $(1+o(1))n/p$ on G.*

As the algorithm's name suggests, we will try and employ techniques inspired by Pósa's *rotation-extension*, which were introduced by Pósa in 1976 [13] in his research of Hamiltonicity in random graphs. Informally put, *rotation-extension* is a technique which under certain conditions allows one to gradually extend paths or cycles in a graph, by finding (through a process usually referred to as

a rotation) a large number of pairs of vertices, such that the existence of an edge between any of these pairs enables one to get a longer path or cycle (an extension) using this edge.
Similarly to the previous results, we will define *CRE* by aligning three algorithms, each calling the next one in case of failure. In essence, the three algorithms will be:

- *CRE1* – A simple greedy algorithm, tasked with optimizing the expected time complexity.
- *CRE2* – The main algorithm, tasked with finding a Hamilton cycle in polynomial time in all but an exponentially small fraction of the probability space.
- *CRE3* – An exponential running time algorithm tasked with finding a Hamilton cycle in the graph when the previous two algorithms failed. This algorithm is identical to *HPA3*.

In Section 2 we present some preliminaries. In Section 3 we present the *CRE* algorithm, and prove its correctness. In Section 4 we prove that the expected running time of *CRE* is $(1+o(1))n/p$. In Section 5 we add some concluding remarks.

2 Preliminaries

In this section we provide several definitions and results to be used in the following sections.
Throughout the paper, it is assumed that all logarithmic functions are in the natural base, unless explicitly stated otherwise.
We suppress the rounding notation occasionally to simplify the presentation.
The following standard graph theoretic notations will be used:

- $N_G(U)$: the external neighbourhood of a vertex subset U in the graph G, i.e.

$$N_G(U) = \{v \in V(G) \setminus U : \ v \text{ has a neighbour in } U\}.$$

- $e_G(U)$: the number of edges spanned by a vertex subset U in a graph G. This will sometimes be abbreviated as $e(U)$, when the identity of G is clear from the context.
- $e_G(U,W)$: the number of edges of G between the two disjoint vertex sets U, W. This will sometimes be abbreviated as $e(U,W)$ when G is clear from the context.

Furthermore, given a cycle or a path S in a graph, with some orientation, we denote:

- S^{-1}: the cycle composed of the vertices and edges of S, but with the opposite orientation.
- $s_S(v)$: the successor of a vertex $v \in S$ on S, according to the given orientation. When the identity of the cycle is clear, we will write $s(v)$.
- $s_S(U)$: the set of successors $\{s_S(u) : \ u \in U\}$. When the identity of the cycle is clear, we will write $s(U)$.

- $p_S(v)$: the predecessor of a vertex $v \in S$ on S, according to the given orientation. When the identity of the cycle is clear, we will write $p(v)$.
- $p_S(U)$: the set of predecessors $\{p_S(u) : u \in U\}$. When the identity of the cycle is clear, we will write $p(U)$.
- $S(v \to u)$: the path $\left(v, s_S(v), s_S^2(v), ..., p_S(u), u\right) \subseteq S$.

Gearing towards our concrete setting of a graph G distributed according to $G(n,p)$ with $p \geq 70n^{-\frac{1}{2}}$, given a graph G, we will define the set of vertices with small degree (with regards to the expected degree) in G:

Definition 1. *Let G be a graph on n vertices. The set $SMALL(G)$ is defined as*

$$SMALL(G) := \{v \in V(G) \mid d(v) < 40\sqrt{n}\}.$$

We shall also make use of the following definition:

Definition 2. *Let $\Gamma = (X \cup Y, E)$ be a bipartite graph. An edge subset $M \subseteq E(\Gamma)$ is called a* ≤ 2-matching *from X to Y if each vertex of X is incident to at most 2 edges in M, and each vertex of Y is incident to at most one edge in M. A* maximum ≤ 2-matching *in Γ is a* ≤ 2-matching *with the maximum possible number of edges.*

We note that given a bipartite graph $\Gamma = (X \cup Y, E)$, a *maximum* ≤ 2*-matching* from X to Y can be found in time $(|X| + |Y|)^{O(1)}$ by using the *MaxFlow* algorithm.

For some of our probabilistic bounds, we will use the following standard result throughout the paper:

Lemma 2.1. (Chernoff bound for binomial tails, see e.g. [11]) *Let $X \sim Bin(n,p)$. Then for every $\delta > 0$, $Pr[X < np - \delta] \leq \exp\left(-\frac{\delta^2}{2np}\right)$.*

3 The *CRE* algorithm

We now present the three components of the *CRE* algorithm, and prove that they are sound. Recall that each component can either fail or return a result, which is either a Hamilton cycle in the input graph or a declaration that there is none. The *CRE* algorithm itself will be:
$CRE(G)$:

If $CRE1(G)$ did not fail, return the result of $CRE1(G)$. Otherwise:
If $CRE2(G)$ did not fail, return the result of $CRE2(G)$. Otherwise:
Return the result of $CRE3(G)$.

3.1 *CRE1*

We present the algorithm *CRE1*. This algorithm will be a greedy algorithm, tasked with optimizing the expected running time. As such, we aim for it to have the following properties, whenever $p \geq 70n^{-\frac{1}{2}}$:

- $\mathbb{E}\left[T_{CRE1}(G)\right] = (1 + o(1))n/p$;
- $Pr[CRE1 \text{ returns } \text{“}failure\text{”}] \cdot \mathbb{E}\left[T_{CRE2}(G)\right] = o(n/p)$.

In the algorithm description we will assume that $V(G) = [n]$.

The *CRE1* algorithm description:

Step 1. Attempt to construct a path P_1 in $G\left([n/2]\right)$ by greedily querying for a neighbour of the current last vertex in the path from outside the path, until the path's end vertex does not have any neighbours among the remaining vertices. If $\frac{n}{2} - |P_1| > \sqrt{n}\log n$, return *"Failure"*. Denote this path by $P_1 = (v_1, ..., v_{n/2-n_1})$, with $n_1 = |[n/2] \setminus P_1|$.
Attempt to construct a path P_2 in $G\left([n/2+1, n]\right)$ in the same manner, and return *"Failure"* if $\frac{n}{2} - |P_2| > \sqrt{n}\log n$. Denote $P_2 = (u_1, ..., u_{n/2-n_2})$.

Step 2. Find indices i, j, k, l with minimal $i+j+k+l$, such that $(v_i, u_{n/2-n_2-j}), (v_{n/2-n_1-k}, u_l) \in E(G)$. If $i + j + k + l > \sqrt{n}\log n$, return *"Failure"*. Otherwise, denote by S_0 the cycle:

$$S_0 := P_1(v_i \to v_{n/2-n_1-k}) \cup \{(v_{n/2-n_1-k}, u_l)\} \cup P_2(u_l \to u_{n/2-n_2-j}) \cup \{(v_i, u_{n/2-n_2-j})\}.$$

Step 3. Initialize $i = 0$, and repeat the following loop until no vertices are left outside the cycle S_i. Choose some vertex $v \notin S_i$. For ease of description we will assume that $v \in [n/2]$. In the complementing case, the description is completely symmetrical, replacing P_2 with P_1, n_2 with n_1 and so on.
Create a set $X = \{x_1, ..., x_{\sqrt[3]{n}}\}$ of neighbours of v on $(P_2 \cap S_i) \setminus \{u_{n/2-n_2-j}\}$ that have not been used in this step, with $z := x_{\sqrt[3]{n}}$ being the maximal one with respect to P_2. Return *"failure"* if no such $\sqrt[3]{n}$ vertices exist. Otherwise, create a set $Y = \{y_1, ..., y_{\sqrt[3]{n}}\}$ of neighbours of $s_{S_i}(z)$ on $(P_1 \cap S_i) \setminus \{v_i\}$. Return *"failure"* if no such $\sqrt[3]{n}$ vertices exist. Finally, find a pair $x \in X \setminus \{z\}, y \in Y$ such that $(s_{S_i}(x), p_{S_i}(y)) \in E(G)$. If no such pair exists, return *"failure"*. Otherwise, set

$$\begin{aligned} S_{i+1} := \quad & \{(v, x)\} \cup S_i^{-1}(x \to y) \cup \{(y, s(z))\} \cup S_i(s(z) \to p(y)) \cup \{(p(y), s(x))\} \\ & \cup S_i(s(x) \to z) \cup \{(z, v)\}; \end{aligned}$$

$i := i + 1$.

3.2 *CRE2*

We present a description of *CRE2*, followed by a proof that the algorithm is sound, that is, if *CRE2* does not fail on a graph G then it returns a Hamilton cycle that is a subgraph of G if and only if G is Hamiltonian.

The *CRE2* algorithm description:

Step 1. Determine $SMALL(G)$ (see Def. 1) by going over all vertices and checking their degrees in G. If the resulting set is larger than $2\sqrt{n}$, return *"Failure"*.

Step 2. Find a *maximum* ≤ 2*-matching* M in G from $SMALL(G)$ to $V(G) \setminus SMALL(G)$. Denote by U the subset of vertices in $V(G) \setminus SMALL(G)$ that have degree 1 in M. If $|U| \leq |SMALL(G)|$, add arbitrary vertices to U until it is of size $|SMALL(G)| + 1$.

Step 3. Using the dynamic programming algorithm (*HPA3*, see description in Section 3.3), find a Hamilton cycle in the graph with vertex set $U \cup SMALL(G)$ and edge set $E_G\,(U \cup SMALL(G)) \cup (U \times U)$. If no such cycle exists, determine that G is not Hamiltonian. Otherwise, denote this cycle by C.
Let $NE = (U \times U) \cap C \setminus E(G)$, let $|NE| = r$, and denote the members of NE by $\{e_1, ..., e_r\}$.

Step 4. For each $1 \leq j \leq r$ find a path P_j of length at most 4 connecting the two vertices of e_j, with all of its internal vertices in $G \setminus \left(\bigcup_{k=1}^{j-1} P_k \cup SMALL(G) \cup U \right)$, using *BFS*. If for some j no such path exists, return *"Failure"*. Otherwise, set $i = 0$ and denote the resulting cycle by $S_0 = \left(C \cup \bigcup_{j=1}^{r} P_j \right) \setminus NE$.

Step 5. Attempt to add at least one vertex of $V(G) \setminus V(S_i)$ to S_i by doing the following:
Using BFS, determine all connected components of $G \setminus S_i$, and denote by V_i a largest connected component. If $|S_i| \geq 0.99n$ and $|V_i| \leq 15\sqrt{n}$, go to *Step 6*. Otherwise, choose an arbitrary orientation to S_i and let $U_i := s(N_G(V_i) \cap S_i)$. If U_i is an independent set, return *"Failure"*. Otherwise, let (u, w) be an edge in U_i, let $u' = p(u), w' = p(w)$ and let P be a path, with all its internal vertices in V_i, connecting u' to w' (this path was uncovered in the BFS stage). Without loss of generality, u precedes w on S_i. Set S_{i+1} to be:

$$S_{i+1} = S_i(w \to u') \cup P \cup S_i^{-1}(w' \to u) \cup \{(u, w)\}.$$

Set $i = i + 1$, and return to *Step 5*.

Step 6. While there is some vertex $v \in V(G) \setminus V(S_i)$, attempt to add it to S_i by exhaustively searching for two vertices $u, w \in N_G(v) \cap S_i$, a set $E_1 \subseteq E(S_i)$ of size at most 4, and a set

$E_2 \subseteq E_G(V(S_i)) \setminus E(S_i)$ of size $|E_1| - 1$, such that $S_{i+1} := (S_i \setminus E_1) \cup E_2 \cup \{(u, v), (v, w)\}$ is a cycle of size $|S_i| + 1$. If no such u, w, E_1, E_2 exist, return *"failure"*.

Lemma 3.1. *If G is a graph such that* CRE2 *does not result in failure when applied to G, then* CRE2 *returns a Hamilton cycle if and only if G is Hamiltonian. Furthermore, if* CRE2 *returns a Hamilton cycle then it is a subgraph of G.*

Proof. In each step $E(S_i) \subseteq E(G)$ and $S_i \subsetneq S_{i+1}$. So it is clear that if the algorithm returns a Hamilton cycle then it is indeed a Hamilton cycle contained in G.
The complementing case is *CRE2* declaring that G is not Hamiltonian. This can only occur in Step 3, if the algorithm failed to find a Hamilton cycle in the graph consisting of vertices $SMALL(G) \cup U$ and edges $E_G(SMALL(G) \cup U) \cup (U \times U)$, which we will denote by H. Since the dynamic programming algorithm was used to find such a cycle, failure to find one means that it does not exist in H, so it remains to be shown that if G is Hamiltonian then H must also be Hamiltonian. We provide a proof of this due to Thomason [14].
Let G^* denote the graph obtained by adding to G all the non-edges with both vertices in $G \setminus SMALL(G)$. Assume that G is Hamiltonian. Then G^* must also be Hamiltonian.
For some Hamilton cycle C, define its *kernel set* to be the edge subset $C \setminus E_{G^*}(V(G) \setminus SMALL(G))$. The kernel set of a Hamilton cycle consists of a set of disjoint paths in $G \setminus E_G(V(G) \setminus SMALL(G))$, containing between them all of $SMALL(G)$, whose endvertices lie in $V(G) \setminus SMALL(G)$.
Let C be a Hamilton cycle in G^* such that the number of edges from M contained in its kernel set is maximised. Denote $SMALL(G) = W_0 \uplus W_1 \uplus W_2$, where W_i is the subset of $SMALL(G)$ joined by i edges of the kernel set to $V(G) \setminus SMALL(G)$. Let $K \subseteq V(G) \setminus SMALL(G)$ be the set of vertices joined by the kernel set to $SMALL(G)$. Then any vertex in W_i matches to at most $2 - i$ vertices in $U \setminus K$, for otherwise if $x \in W_i$ and $(x, y) \in M$, where $y \notin K$, we can remove a kernel set edge from x, replace it with (x, y), and create a new kernel set (of another Hamilton cycle C') with more edges from M in it. Now, for each vertex in K, choose an edge of the kernel set incident to it arbitrarily. Then a vertex of W_i is incident with at most i of these edges. So these edges, along with the edge set $M \cap (SMALL(G) \times (U \setminus K))$, together form a ≤ 2-matching of order $|U \cup K|$. Since the largest ≤ 2-matching has order exactly $|U|$, we see that $K \subseteq U$. It now follows from the definition of a kernel set that we can construct a Hamilton cycle in H, as claimed.

□

3.3 *CRE3*

The final part of *CRE* is *CRE3*, an algorithm with the following desired properties:

- The time complexity of *CRE3* is $2^{2n} \cdot n^{O(1)}$;
- The space complexity of *CRE3* is linear in n;

- The result of *CRE3* is either a Hamilton cycle contained in the input graph, or a declaration that the graph is not Hamiltonian if the input graph contains none.

Luckily, such an algorithm already exists — the algorithm *HPA3* presented by Gurevich and Shelah in [9]. For completeness we give a brief description of the algorithm. For proof of the properties, see the original paper. We note that, as mentioned in Section 1, an algorithm with time complexity $O\left(2^n \cdot n^2\right)$ is known. The downside of this algorithm is that it also has exponential space complexity. This is not a very big issue for us, since our interests in this paper lie exclusively in time complexity, but since we can get a similar algorithm, but with linear space, with its time complexity still sufficiently small for our purposes, this is the one we chose.

The algorithm *HPA3*, given a graph G and two vertices $s, t \in V(G)$, finds a Hamilton path in G from s to t. First we note that converting this algorithm into an algorithm for finding a Hamilton cycle is very simple: choose an arbitrary vertex in G, say s, and iterate $HPA3(G \setminus (s,t), s, t)$ over all $t \in N_G(s)$. If for some t a Hamilton $s-t$ path P is found then $P \cup (s,t)$ is a Hamilton cycle in G. If all iterations fail, then surely G cannot be Hamiltonian.

HPA3 is defined recursively, as follows:

$HPA3(G, s, t)$:

If $V(G) = \{s, t\}$*, return* (s, t) *if it is an edge, and* **"No such path"** if it is not an edge. *Otherwise: For all* $c \in V(G) \setminus \{s, t\}$ *and for all* $A \subseteq V(G) \setminus \{s, t, c\}$ *of size* $\lfloor \frac{n-3}{2} \rfloor$*:*

If $HPA3(A, s, c)$ *and* $HPA3(G \setminus A, c, t)$ *are successful, return* $HPA3(A, s, c) \cup HPA3(G \setminus A, c, t)$*; otherwise, continue.*

If loop failed, return **"No such path".**

4 Expected time complexity of *CRE*

In this section we aim to prove that the algorithm described in Section 3 meets the time complexity goals we had set, that is: if $p \geq 70n^{-\frac{1}{2}}$, then the expected running time over $G(n,p)$ is $(1+o(1))n/p$. Since

$$\begin{aligned} \mathbb{E}\left[T_{CRE}(G)\right] \quad \leq \quad & \mathbb{E}\left[T_{CRE1}(G)\right] + Pr[CRE1 \text{ fails}] \cdot \mathbb{E}\left[T_{CRE2}(G) \,|\, CRE1 \text{ fails}\right] \\ & + Pr[CRE2 \text{ fails}] \cdot \mathbb{E}\left[T_{CRE3}(G)\right], \end{aligned}$$

it is sufficient to prove that the following hold:

- $\mathbb{E}\left[T_{CRE1}(G)\right] = (1 + o(1))n/p$;
- $Pr[CRE1 \text{ fails}] \cdot \mathbb{E}\left[T_{CRE2}(G) \,|\, CRE1 \text{ fails}\right] = o(n/p)$;
- The probability that *CRE2* returns *"failure"* is $2^{-2n} \cdot n^{-\omega(1)}$;
- The running time of *CRE3* is $2^{2n} \cdot n^{O(1)}$.

A proof of the last point is provided in [9]. We now provide proofs for the other three points.

4.1 Expected running time of *CRE1*

Lemma 4.1. *If $p \geq 70n^{-\frac{1}{2}}$, $G \sim G(n,p)$, then $\mathbb{E}[T_{CRE1}(G)] = (1+o(1))n/p$.*

Proof. The expected running time of *CRE1* is the sum of the expected running times of its three steps.

- In Step one *CRE1* samples edges, until it reaches at most $n-2$ successes, which means that the expected time of this step is at most $(n-2)/p$;
- In Step 2 *CRE1* samples edges until it finds two existing edges. So the expected running time of this step is $2/p$;
- In Step 3 *CRE1* repeats a loop at most $\sqrt{n}\log n$ times. In each time, it samples edges until it finds $2\sqrt[3]{n}+1$ existing ones. So the expected running time of this step is at most $\sqrt{n}\log n \cdot (2\sqrt[3]{n}+1)/p = o(n/p)$.

Overall, we get the desired sum of $(1+o(1))n/p$.

□

4.2 Probability of failure of *CRE1*

Lemma 4.2. *Let $p \geq 70n^{-\frac{1}{2}}$ and let $G \sim G(n,p)$. Then the probability that $CRE1(G)$ returns the result* "failure" *is $o(n^{-60})$.*

Proof. We note that since no edge is sampled twice during the run of *CRE1*, all the possible events that lead to failure are independent. We bound from above the probability of each of these events occurring.

1. *CRE1* fails if at some point in Step 1 the last vertex in P_1 has no neighbours in the set $[n/2] \setminus P_1$, and if at that point this set is larger than $\sqrt{n}\log n$. The probability of this occurring is at most the probability that among $\frac{n}{2}$ independent random variables distributed $Bin(\sqrt{n}\log n, p)$ at least one is equal to zero. We bound this probability by applying the union bound:
$$\begin{aligned} Pr[n_1 \geq \sqrt{n}\log n] &\leq 0.5n \cdot (1-p)^{\sqrt{n}\log n} \\ &\leq 0.5n\exp(-70\log n) \\ &= o(n^{-60}). \end{aligned}$$
2. $$Pr[n_2 \geq \sqrt{n}\log n] = Pr[n_1 \geq \sqrt{n}\log n] = o(n^{-60}).$$

3. Step 2 results in failure if the minimal indices i,j,k,l for which $\left(v_i, u_{n/2-n_2-j}\right), \left(v_{n/2-n_1-k}, u_l\right)$ are in $E(G)$ satisfy $i+j+k+l > \sqrt{n}\log n$, and in particular $i+j > 0.5\sqrt{n}\log n$ or $k+l > 0.5\sqrt{n}\log n$. There are $\binom{0.5\sqrt{n}\log n}{2} \geq 0.1n\log^2 n$ pairs i,j (or k,l) with $i+j \leq 0.5\sqrt{n}\log n$, for which an edge query resulted in failure. Applying the union bound we get

$$\begin{aligned} Pr[i+j+k+l > \sqrt{n}\log n] &\leq 2Pr[i+j > 0.5\sqrt{n}\log n] \\ &\leq (1-p)^{0.1n\log^2 n} = n^{-\omega(1)}. \end{aligned}$$

4. If Step 3 resulted in failure, say in the m'th iteration, then there was some vertex v outside of S_m such that one of the following happened:

   (a) v did not have $\sqrt[3]{n}$ neighbours in (wlog) $(P_2 \cap S_m) \setminus \{u_{n/2-n_2-j}\}$ that have not been used in iterations 0 to $i-1$;

   (b) $s_{S_m}(z)$ did not have $\sqrt[3]{n}$ neighbours in $(P_1 \cap S_m) \setminus \{v_i\}$;

   (c) $s(X)$ and $p(Y)$ did not have any edge between them.

   Since up to the m'th iteration, at most $\sqrt{n}\log n \cdot \sqrt[3]{n} = o(n)$ vertices of S_m have been used, the probability of *(a)* and *(b)* is at most the probability that $Bin(n/6,p) < \sqrt[3]{n}$. So:

$$\begin{aligned} Pr[\text{Step 3 failed}] &\leq n \cdot \left(2 \cdot Pr\left[Bin(n/6,p) < \sqrt[3]{n}\right] + (1-p)^{\sqrt[3]{n}(\sqrt[3]{n}-1)}\right) \\ &\leq n \cdot \left(\exp\left(-\Omega(\sqrt{n})\right) + \exp\left(-\Omega(\sqrt[6]{n})\right)\right) = n^{-\omega(1)}. \end{aligned}$$

So all of the events that lead to failure have probability $o(n^{-60})$, and therefore the probability of failure is also $o(n^{-60})$, as we have set out to prove.

□

4.3 Expected running time of *CRE2*

Lemma 4.3. *Let $p = p(n) \geq 70n^{-\frac{1}{2}}$. Then $Pr[CRE1\ fails] \cdot \mathbb{E}\left[T_{CRE2}(G) \,|\, CRE1\ fails\right] = O(1)$, where the input to both algorithms is distributed according to $G \sim G(n,p)$.*

Proof. Denote $Pr[CRE1 \text{ fails}] := p_1$. Except for Step 3, all steps of *CRE2* have time complexity at most $O(n^5)$, regardless of the input graph. As for Step 3, since $|U \cup SMALL(G)| \leq 3|SMALL(G)|$, the expected runtime of this step (assuming we reach it) is

$$\begin{aligned} \mathbb{E}\left[T_{Step\ 3}(G) \,|\, CRE1 \text{ fails}\right] &= \sum_{k=1}^{2\sqrt{n}} k^{O(1)} 2^{6k} \cdot Pr\left[|SMALL(G)| = k \,|\, CRE1 \text{ fails}\right] \\ &\leq {p_1}^{-1} \cdot \sum_{k=1}^{2\sqrt{n}} k^{O(1)} 2^{6k} \cdot Pr\left[|SMALL(G)| = k\right]. \end{aligned}$$

We bound each term from above, using the Chernoff bound (Lemma 2.1)

$$\begin{aligned} k^{O(1)} 2^{6k} \cdot Pr\left[|SMALL(G)| = k\right] &\leq k^{O(1)} 2^{6k} \cdot \binom{n}{k} \cdot Pr\left[Bin(k(n-k),p) \leq \tfrac{3}{4}knp\right] \\ &\leq \exp\left(O(\log k) + 6k + k\log n - \Omega(knp)\right) = o\left(n^{-1}\right), \end{aligned}$$

hence the value of the entire sum above is at most $o(1)$.
So overall

$$Pr[CRE1 \text{ fails}] \cdot \mathbb{E}\left[T_{CRE2}(G) \,|\, CRE1 \text{ fails}\right] = p_1 \cdot O\left(n^5 + {p_1}^{-1}\right) = O(1).$$

□

4.4 Probability of failure of *CRE2*

Let $G \sim G(n,p)$, where $p = p(n) \geq 70n^{-\frac{1}{2}}$.
We will call an event A *rare* if $Pr[A] = 2^{-2n} \cdot n^{-\omega(1)}$. Our goal is to prove that $CRE2(G)$ resulting in *failure* is a rare event. We aim to do this by presenting a graph property (P) such that:

- $G \notin (P)$ is rare;
- If $G \in (P)$ then *CRE2* deterministically either finds a Hamilton cycle or determines that the graph is not Hamiltonian.

Define the graph property (P) as follows:

$$\forall U, W \subseteq V(G) \textit{ disjoint subsets } : e(U,W) > |U| \cdot |W| \cdot p\left(1 - \sqrt{\frac{n^{1.5}}{10|U| \cdot |W|}}\right).$$

(In particular, if $|U| \cdot |W| \geq \frac{n^{1.5}}{10}$ then $e(U,W) \geq 1$.)

Lemma 4.4. *If $p = p(n) \geq 70n^{-\frac{1}{2}}$ and $G \sim G(n,p)$, then $G \notin (P)$ is rare.*

Proof. We bound from above the probability that $G \notin (P)$.
Let $U, W \subseteq V(G)$ be two disjoint sets, and assume that $|U| \cdot |W| \geq \frac{n^{1.5}}{10}$. By the Chernoff bound (Lemma 2.1), the probability of $e(U,W) \leq |U| \cdot |W| \cdot p\left(1 - \sqrt{\frac{n^{1.5}}{10|U| \cdot |W|}}\right)$ is at most

$$Pr\left[Bin\left(|U| \cdot |W|, p\right) \leq |U| \cdot |W| \cdot p\left(1 - \sqrt{\frac{n^{1.5}}{10|U| \cdot |W|}}\right)\right] \leq \exp\left(-\frac{1}{20} \cdot n^{1.5}p\right) \leq e^{-3.5n}.$$

Finally, by the union bound we get that the probability that exist such U, W is at most $3^n \cdot e^{-3.5n} = 2^{-2n} \cdot n^{-\omega(1)}$, as desired. □

In order to prove that *CRE2* does not result in *"failure"* on an input graph G satisfying (P) for $p = p(n) \geq 70n^{-\frac{1}{2}}$, we will show that none of the four stages that may result in *"failure"* does so on such an input.
In the following lemmas it is assumed, without stating explicitly, that $p(n) \geq 70n^{-\frac{1}{2}}$.

Lemma 4.5. *Let G be a graph on n vertices satisfying (P). Then* Step 1 *does not return* "Failure" *on input G.*

Proof. *CRE2* fails this step if and only if $|SMALL(G)| \geq 2\sqrt{n}$. Let $A \subseteq SMALL(G)$ be some subset of size $2\sqrt{n}$. So A and $V(G) \setminus A$ are two disjoint subsets with $|A| \cdot |V(G) \setminus A| \geq 1.9n^{1.5}$, but

$$e\left(A, V(G) \setminus A\right) \leq 40\sqrt{n}|A| \leq \left(1 - \frac{1}{\sqrt{19}}\right) \cdot |A| \cdot |V(G) \setminus A| \cdot p,$$

a contradiction to G satisfying (P). □

Lemma 4.6. *Let G be a graph on n vertices satisfying (P). Then* Step 4 *does not return* "Failure" *on input G.*

Proof. Say we failed to find a path of length at most 4 between the vertices of some non-edge $e_i := (u_1, u_2) \in U \times U$ in the graph $H_i := G \setminus \left(\bigcup_{j=1}^{i-1} P_j \cup SMALL(G) \cup U\right)$. Since $u_1, u_2 \notin SMALL(G)$, it holds that

$$|N_{H_i}(u_1)|, |N_{H_i}(u_2)| \geq 40\sqrt{n} - 6 \cdot |SMALL(G)| \geq 25\sqrt{n}.$$

Let $D_2(G, v)$ denote the set of vertices in a graph G of distance at most 2 from a vertex v. Because there is no path of length at most 4, the sets $D_2(H_i, u_1), D_2(H_i, u_2)$ do not intersect each other, which means that one of them, WLOG $D_2(H_i, u_1)$, is of size at most $\frac{1}{2}n$. But then we have

$$|N_{H_i}(u_1)| \cdot |H_i \setminus (D_2(H_i, u_1) \cup \{u_1\})| \geq 25\sqrt{n} \cdot \left(n - 12\sqrt{n} - \frac{1}{2}n - 1\right) \geq 10n^{1.5},$$

$$e\left(N_{H_i}(u_1), H_i \setminus (N_{H_i}(u_1) \cup D_2(H_i, u_1) \cup \{u_1\})\right) = 0,$$

which means $G \notin (P)$, a contradiction. □

Lemma 4.7. *Let G be a graph on n vertices satisfying* (P). *Then* Step 5 *does not return* "Failure" *on input G.*

Proof. Say we failed at some time i, that is: the constructed vertex set U_i is an independent set. Recall that U_i is the set of successors along S_i of vertices in $N_G(V_i) \cap S_i$, where V_i is a maximum sized connected component of $G \setminus S_i$. Let $W_i = N_G(V_i) \cap S_i$. Consider the following cases:

1. $|U_i| \geq n^{\frac{3}{4}}$. Let $A_1, A_2 \subseteq U_i$ be two disjoint subsets of size $\frac{1}{2}n^{\frac{3}{4}}$. So $|A_1| \cdot |A_2| = \frac{1}{4}n^{1.5}$, but $e(A_1, A_2) = 0$, a contradiction.

2. $|V_i| > n - 30\sqrt{n}$. Observe two facts:

   - By Def. 1, since $|V(G) \setminus V_i| < 30\sqrt{n} < 40\sqrt{n}$, we get that $\forall v \in S_i \setminus SMALL(G)$: $N_G(v) \cap V_i \neq \emptyset$;

- Since $|SMALL(G)| < \frac{1}{2}|S_0| \leq \frac{1}{2}|S_i|$, there are two vertices $w_1, w_2 \in S_i \setminus SMALL(G)$ such that $w_1 = s_{S_i}(w_2)$.

So w_1, w_2 belong to W_i, and their successors are connected by an edge, which means that the algorithm could not have failed.

3. $15\sqrt{n} \leq |V_i| \leq n - 30\sqrt{n}$. Observe that if the algorithm failed then $|W_i| = |U_i| \leq \min\{\frac{1}{2}|S_i|, n^{3/4}\}$, and therefore we have

   - $|V_i| + |V(G) \setminus (V_i \cup W_i)| \geq n - n^{\frac{3}{4}}$;
   - $|V_i| \geq 15\sqrt{n}$;
   - $|V(G) \setminus (V_i \cup W_i)| = |V(G) \setminus V_i| - |W_i| \geq |V(G) \setminus V_i| - \frac{1}{2}|S_i| \geq \frac{1}{2}|V(G) \setminus V_i| \geq 15\sqrt{n}$.

   So V_i and $V(G) \setminus (V_i \cup W_i)$ are two sets, with $|V_i| \cdot |V(G) \setminus (V_i \cup W_i)| \geq 10n^{1.5}$, but $e(V_i, V(G) \setminus (V_i \cup W_i)) = 0$, a contradiction to our assumption that $G \in (P)$.

4. $|V_i| \leq 15\sqrt{n},\ |S_i| < 0.99n$. Then all connected components of $G \setminus S_i$ are of size at most $15\sqrt{n}$, and the sum of their sizes is at least $0.01n$. So the vertices of $V(G) \setminus S_i$ can be partitioned into two sets A_1, A_2 such that each one of them is a union of connected components, and $|A_1|, |A_2| \geq n^{\frac{3}{4}}$. But then $|A_1| \cdot |A_2| \geq n^{1.5}$ and $e(A_1, A_2) = 0$, a contradiction.

The complementing case to those already covered is when $|V_i| \leq 15\sqrt{n},\ |S_i| \geq 0.99n$, which can only occur in Stage 6. □

Lemma 4.8. *Let G be a graph on n vertices satisfying* (P). *Then* Step 6 *does not return* "Failure" *on input G.*

Proof. We show that under the assumption that $G \in (P)$, the cycle S_i contains two vertices u, w and two edge subsets E_1, E_2 as described in *Step 6*. Since the algorithm searches for such u, w, E_1, E_2 exhaustively, and only returns *"Failure"* upon failing the search, this means that if $G \in (P)$ the algorithm does not fail.

Recall that in this stage we can assume that $|S_i| \geq 0.99n$ and that all connected components of $G \setminus S_i$ are of size at most $15\sqrt{n}$. It follows that for every $v \in V(G) \setminus S_i$ we have $|N_G(v) \cap S_i| \geq d_G(v) - |V_i| \geq 20\sqrt{n}$. Observe that if $|N_G(v) \cap S_i| > \frac{1}{2}n$ then v has two neighbours adjacent on S_i, say u, w, so setting $E_1 = (u, w),\ E_2 = \emptyset$ results in a cycle as desired, so we can assume that $|N_G(v) \cap S_i| \leq \frac{1}{2}n$.

Let $U_i := p(N_G(v) \cap S_i)$. Since $|U_i|, |S_i \setminus U_i| \geq 20\sqrt{n}$ and $|U_i| + |S_i \setminus U_i| \geq 0.99n$, we get that $|U_i| \cdot |S_i \setminus U_i| \geq 10n^{1.5}$, and therefore $e(U_i, S_i \setminus U_i) \geq 0.9p|U_i| \cdot |S_i \setminus U_i| \geq 0.4|U_i|np$. It follows that there is some $u \in N_G(v) \cap S_i$ such that $d_{S_i}(p(u)) \geq 0.4np \geq 20\sqrt{n}$. Denote $t_0 := p(u)$, and let Q be the path $\{v\} \cup S_i(u \to t_0)$.

Define the following three special vertices on Q:

- c_v : a vertex on Q such that $|N_{Q(v\to c_v)}(v)| = \lfloor \frac{1}{2}|N_Q(v)| \rfloor$;
- c_{t_0} : a vertex on Q such that $|N_{Q(c_{t_0}\to t_0)}(t_0)| = \lfloor \frac{1}{2}|N_Q(t_0)| \rfloor$;
- c : a vertex on Q such that $|Q(v \to c)| = \lfloor \frac{1}{2}|Q| \rfloor$.

We will assume that c_v and c precede c_{t_0} on Q, and remark that the proof is quite similar for the complementing cases, in which c_{t_0} precedes one or both of c_v, c, with some minor changes required to some of the definitions down the line.
Denote: $Q_1 := Q(v \to c_v),\ Q_2 := Q(c_{t_0} \to t_0),\ Q_3 := Q(v \to c)$.
We now aim to show that E_1, E_2, w as required exist in the graph, with respect to the already chosen u, by using rotations and extensions.
Let W_i be the set $N_{Q_1}(v)$ and T_i the set $s_Q(N_{Q_2}(t_0))$. By our choices of v, t_0, c_v, c_{t_0} we know that $|W_i|, |T_i| \geq 10\sqrt{n}$. Now, construct the set O_i as follows:
For each vertex $x \in W_i$ and for each $y \in N_{Q_3}(p_Q(x)) \setminus \{x\}$ add $s_Q(y)$ to O_i if $y \in Q(v \to x)$ and add $p_Q(y)$ to O_i if $y \in Q(x \to c)$.

Claim 4.1. *The size $|O_i|$ is at least $0.2n$.*

Proof. By our construction, $|O_i| \geq |N_{Q_3}(p_Q(W_i))| - |W_i|$. If $|O_i| < 0.2n$ then $|Q_3 \setminus N_{Q_3}(p_Q(W_i))| \geq 0.25n - |W_i|$, and $p_Q(W_i),\ Q_3 \setminus N_{Q_3}(p_Q(W_i))$ are two sets that have no edges between them, but the product of their sizes is at least $2n^{1.5}$, a contradiction. □

Claim 4.2. *There is an edge between O_i and T_i.*

Proof. The two sets are disjoint, and $|O_i| \cdot |T_i| \geq 2n^{1.5}$. □

Let $s \in O_i,\ t \in T_i$ be such that $(s,t) \in E(G)$, and let $w \in W_i$ be a vertex that caused s to be added to O_i. Finally, define:

- $E_1 := \{(u, t_0),\ (p_Q(w), w),\ (s, s_Q(s)),\ (p_Q(t), t)\}$;
- $E_2 := \{(p_Q(w), s_Q(s)),\ (p_Q(t)), t_0),\ (s, t)\}$.

Then E_1, E_2, u, w are as required by the algorithm (see Fig. 1 for illustration).

□

5 Concluding remarks

To summarise, we have presented an algorithm *CRE* which is comprised of three aligned algorithms, in the spirit of previous results, and utilises rotations and extensions in order to find a Hamilton cycle in a graph, and proved that its expected running time on a random graph $G \sim G(n,p)$ is optimal, for $p \geq 70n^{-\frac{1}{2}}$.

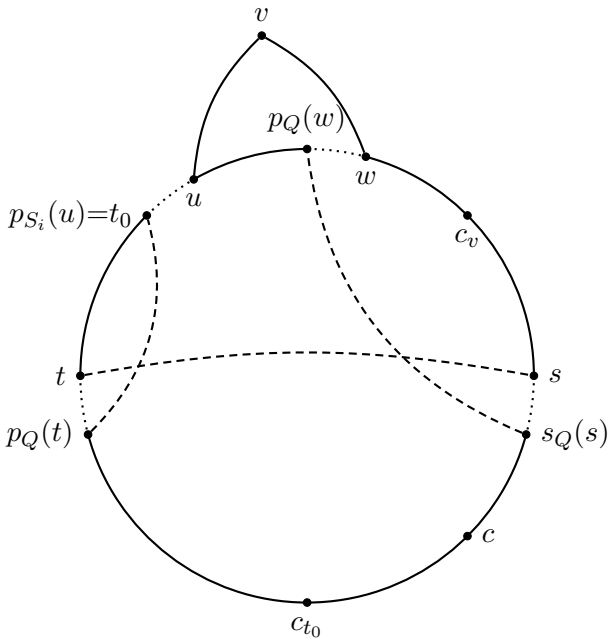

Figure 1: Extention of cycle S_i (oriented clockwise) to cycle S_{i+1} that includes v, by removing the edges of E_1 (dotted) and adding the edges of E_2 (dashed) and $(v,u),(v,w)$.

We note that even if we make changes to some parameters in our algorithm, $p = \Omega\left(n^{-\frac{1}{2}}\right)$ seems to be the lowest range of probability for which our expected running time bound works, at least with our current argument. The reason for this is the existence of some bottlenecks along the proof, where smaller orders of magnitude of the edge probability no longer work. Such a bottleneck can be observed, for example, in Step 4 of *CRE2*, where the algorithm tries to connect some set of paths into a cycle that contains them, by finding paths between pairs of endpoints of paths one by one. In our proof we use the fact that the total length of the paths is highly likely to be much smaller than the minimum degree of the vertices at the endpoints of the paths (that is to say that the complement event is rare, i.e., has probability $2^{2n} \cdot n^{-\omega(1)}$). This is due to the fact that, on the one hand, all of the paths' endpoints have degrees at least comparable to the expected average degree of the graph, since by our construction none of the endpoints are members of $SMALL(G)$ – the set of vertices with very small degrees. On the other hand, the total number of vertices in the union of all the paths is not likely to be very big, since this vertex set contains at most $6 \cdot |SMALL(G)|$ vertices, a size likely to be much smaller than the average degree of the graph for our parameters, as we observed that $SMALL(G)$ is highly likely to be of size much smaller than np. If $p = o\left(n^{-\frac{1}{2}}\right)$, however, then the event "$|SMALL(G)| > np$" has probability $2^{-o(n)}$, and in particular it is no longer rare. In other words, the probability that one of the paths' endpoints has all its neighbours residing in the union of $SMALL(G)$ and previously constructed paths is $2^{-o(n)}$, and the expected runtime of *CRE* might no longer even be polynomial.

And so, we leave it as an open question whether a polynomial expected running time Hamiltonicity algorithm exists for edge probability $p = o\left(n^{-\frac{1}{2}}\right)$.

Acknowledgements. The authors would like to express their thanks to the referees of the paper, and to Samotij Wojtek, for their valuable input towards improving the presentation of our result.

References

[1] M. Ajtai, J. Komlós and E. Szemerédi, *First occurrence of Hamilton cycles in random graphs*, Cycles in graphs '82, North Holland Mathematical Studies 115, North Holland, Amsterdam (1985), 173–178.

[2] R. Bellman, *Dynamic programming treatment of the travelling salesman problem*, Journal of the ACM 9 (1962), 61–63.

[3] B. Bollobás, *The evolution of sparse graphs*, Graph Theory and Combinatorics, Academic Press, London (1984), 35–57.

[4] B. Bollobás, T. Fenner and A. Frieze, *An algorithm for finding Hamilton paths and cycles in random graphs*, Combinatorica 7 (1987), 327–341.

[5] A. Ferber, M. Krivelevich, B. Sudakov and P. Vieira, *Finding Hamilton cycles in random graphs with few queries*, Random Structures & Algorithms 49 (2016), 635–668.

[6] A. Frieze, *Hamilton cycles in random graphs: a bibliography*, arXiv preprint arXiv:1901.07139 (2019).

[7] A. Frieze and C. McDiarmid, *Algorithmic theory of random graphs*, Random Structures & Algorithms 10 (1997), 5–42.

[8] M. Garey, D. Johnson and L. Stockmeyer, *Some simplified NP-complete graph problems*, Theoretical Computer Science 1.3 (1976), 237–267.

[9] Y. Gurevich and S. Shelah, *Expected computation time for Hamiltonian path problem*, SIAM Journal on Computing 16 (1987),486–502.

[10] M. Held and R. Karp. *A dynamic programming approach to sequencing problems*, Journal of the Society for Industrial and Applied Mathematics 10 (1962), 196–210.

[11] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association 58 (1963), 13–30.

[12] J. Komlós and E. Szemerédi, *Limit distributions for the existence of Hamilton circuits in a random graph*, Discrete Mathematics 43 (1983), 55–63.

[13] L. Pósa, *Hamiltonian circuits in random graphs*, Discrete Mathematics 14 (1976), 359–364.

[14] A. Thomason, *A simple linear expected time algorithm for finding a Hamilton path*, Discrete Mathematics 75 (1989), 373–379.