Analog Computer Understanding of Hamiltonian Paths, and a Possible Digitization

Bryce M. Kim

07/07/2016

Abstract

This paper explores finding existence of undirected hamiltonian paths in a graph using lumped/ideal circuits, specifically low-pass filters. While other alternatives are possible, a first-order RC low-pass filter is chosen to describe the process. The paper proposes a way of obtaining time complexity for counting the number of hamiltonian paths in a graph, and then shows that the time complexity of the circuits is around $O(n \log n)$ where n is the number of vertices in a graph. Because analog computation is often undesirable due to several aspects, a possible digitization scheme is proposed in this paper.

1 Introduction: Undirected Hamiltonian Path Existence Problem

Graph $G = (V, E)$ is defined with a set V of vertices along with a set E of undirected edges connecting vertices. We will denote a walk by the following formalism: $a - b - c$ where a, b, c are vertices and $-$ represents edges. Generally, a, b, c will be represented with positive integers.
$n = n_v$ is the cardinality of V, n_e is the cardinality of E. A n-walk is defined to be a walk with n vertices. This is the only class of walks we will have interests in this paper.
We will re-interpret Hamiltonian path existence problem using a $n \times n$ grid and others.

Definition 1.1. The grid contains n vertical columns. Each column contains n vertices, and the vertices in the same column are not connected by wires.

Definition 1.2. All vertices are numbered with positive integers greater than 1.

Definition 1.3. Each wire transmits a voltage signal $f(t)$. For our consideration, location does not matter, so all of our signals are solely function of time. These signals can be transformed into the Fourier transform frequency representation.

Definition 1.4. As part of lumped circuit assumption, we will assume that wires have no time delay. (ideal wire)

Definition 1.5. For each vertex x at column $a > 1$, if vertex y satisfies $(x, y) \in E$ or $(y, x) \in E$, vertex y frequency multiplier (or oscillator, in case of $a-1 = 1$) at column $a-1$ is connected by a wire to the sum operator at vertex x/column a.

Definition 1.6. As we allow self-loops, while $(x, x) \notin E$, vertex x at column $a-1$ is connected by a wire to the sum operator at vertex x/column a.

Definition 1.7. Each vertex x at column 1, the first column, only has an ideal oscillator, transmitting e^{ixt} to wires connected to the second column.

Definition 1.8. A sum operator just sums up the signals transmitted by wires.

Definition 1.9. Each sum operator at vertex x/column a is connected to a frequency multiplier at the same vertex/column, with frequency multiplication factor of x. Frequency multiplier transforms $e^{iw_1 t} + e^{iw_2 t} + ...$ into $e^{ixw_1 t} + e^{ixw_2 t} + ...$

Definition 1.10. At column n, after signals pass through frequency multipliers connected to sum operators, any wire incident from column n is connected to a final sum operator, which produces the final signal $y(t)$.

Thus it is clear that we need $n(n-1)+1$ sum operators (or adders, equivalently) and $n(n-1)$ frequency multipliers for the grid above. The number of wires are dependent on E, but the maximum number of wires required is $n^2(n-1)+n(n-1)+n$, where the last n comes the wires that connect column n to the last sum operator, and $n(n-1)$ comes from the wires that connect a single sum operator to a single frequency multipliers.
The output of the circuit grid defined above is $y(t)$, as mentioned above. Let $V = \{v_1, v_2, .., v_n\}$.

Definition 1.11. The final sum operator, which produces the signal $y(t)$ is connected to the ideal mixer M, which outputs the product of $y(t)$ with e^{-iut} where $u = v_1 v_2 v_3 .. v_n$. In Fourier transform, this is equivalent to converting $Y(\omega)$ with $Y(\omega + u)$, where $Y(\omega)$ is Fourier transform of $y(t)$. Let the output of M be $k(t)$.

From the above, it is clear that Ce^{iut} inside $y(t)$ represents hamiltonian paths, with C representing the number of hamiltonian paths. In $k(t)$, frequency 0 represents hamiltonian paths, as all frequencies are shifted left by u.
Because our chosen low-pass filter will be first-order, we will also pass $k(t)$ to a frequency multiplier that multiplies frequencies by ${v_n}^{4n}$ where v_n is the greatest-numbered vertex, to ensure that the frequencies other than zero frequency parts of $k(t)$ will be sufficiently high frequencies. (Multiplying zero by ${v_n}^{4n}$ is zero) For higher-order filters, like third-order filter, this additional frequency-multiplying process will not be needed. We will call the resulting signal $j(t)$.
As a side note, instead of having input tape in Turing machine, we have to re-wire $n \times n$ grid every time graph input changes. This $n \times n$ grid serves as an input to the system involving a low-pass filter.

1.1 Restriction on vertex indices

However, a close look will reveal that it is required for us to restrict on vertex indices. Hamiltonian u may be decomposed into a product of n numbers that are in V, and yet all these numbers may not be distinct, required for u to represent hamiltonian paths. One simple way to address this problem is by required all vertex indices to be prime numbers. For simplification, assume that $v_1 = 2$ and $v_n = p_n$ where p_k represents kth prime number with $p_1 = 2$. It is known that $p_n < n(\ln n + \ln \ln n)$, shown in Rosser (1941). Thus we only need to check non-exponential number of natural numbers to obtain n prime numbers to be used as indices for vertices.

2 Introduction: Alternative Circuit Formation

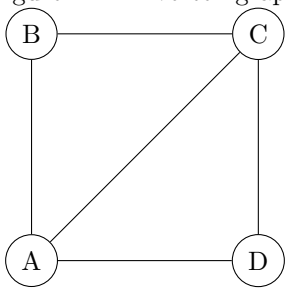
The method using the alternative circuit/grid method can be divided into two parts: the first part is about converting an undirected graph into a Fourier series $f(t) : \mathbb{R} \to \mathbb{C}$. The second part is providing effective low-pass filtering without sacrificing control of transient response in case of a digitization method and simple low-pass filtering in case of an ideal analog case.
The purpose of the the grid part is to set the amplitude A_0 in $f(t) = \sum_{u \in U} A_u e^{iut}$, which is amplitude at frequency 0 to be the number of hamiltonian paths given as n_h, where U is the set of angular frequencies that have non-zero amplitude in Fourier series form.
A vertex-number is assigned to each vertex (if not mentioned explicitly, vertex refers to a vertex-number it is assigned), and in a particular implementation of the method used in this paper, each vertex is $v = n^i$ with $i \in \mathbb{Z}_+$. Each walk with $|V| = n$ vertices, now called n-walks, is given a frequency, which is implemented to be the sum of vertex-numbers of vertices the walk contains. If n-walks share the same set of vertex-numbers, but only the visit order is different (permutation), then they have the same angular frequency.
Let $x(t)$ be the sum of all n-walks w, where each walk is given by e^{iwt} as in the above.
Notice from the above that the angular frequency of each walk is simply the sum of vertex-numbers. And note:

$$(e^{iv_{1,1}t} + e^{iv_{1,2}t})e^{iv_{2,3}t} = e^{i(v_{1,1}+v_{2,3})t} + e^{i(v_{1,2}+v_{2,3})t}$$

$v_{1,1}$ and $v_{1,2}$ can be thought of the first vertex visited for each corresponding n-walk. Both walks share the same second vertex visited, which is $v_{2,3}$. Notice that the first index (1 in $v_{1,2}$, for example) presents the visit order of a walk, while the second index distinguishes actual vertex. Thus, $v_{1,3}$ and $v_{3,3}$ represent the same vertex, but with a different visit order.
The full details will be given in one of the next sections, but for now let us illustrate the principles using the example in Figure 2 and 2:

In Figure 2, because the original graph in 2 is a 4-vertex graph, there are four layers or four depths, labelled with L1,L2,L3,L4. Each layer i contains all

Figure 1: A 4-vertex graph



vertices in a graph. A vertex v in layer i is connected to a vertex w in layer $i+1$ whenever $(v, w) \in E$. This grid procedure effectively simulates an actual walk. At Layer 1 (L1), each vertex v transmits e^{ivt} to the edges, or wires, it is connected with. These edges connect to the vertices at the next layer L2. For other layers, each vertex v sums up all the function it received from the wires starting from the previous layer and multiplies the sum by e^{ivt}, and then transmits the product to the wires that connect v with the vertices of the next layer. The final layer L4 has an additional step, since there are no wires that connect to the next layer in Figure 2. Instead, all results obtained at each vertex at L4 is summed up, which results in $x(t)$.

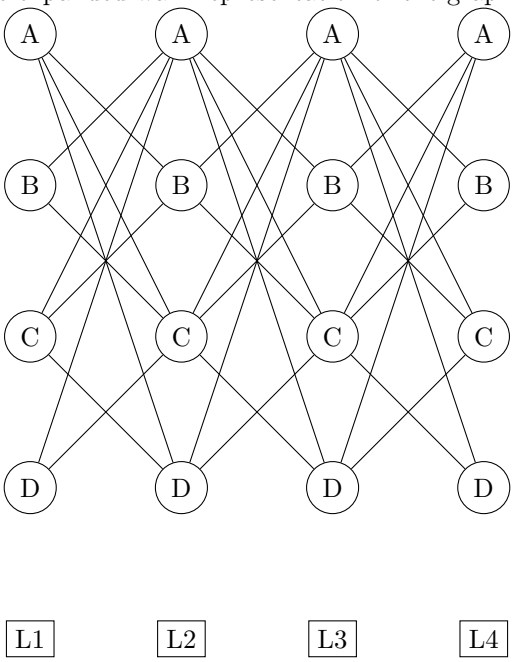
To summarize, the vertices in L1 always act as oscillators, every edge that connects one layer to the next layer acts as a right-directional wire without any transmission delay and the vertices in each layer except L1 work first as a summer and then a multiplier coupled with an oscillator. After the final layer, a summer adds up all results in the vertices in the final layer.

The constructed $x(t)$ does not have zero frequency as a hamiltonian path frequency and thus multiplication by e^{-iht}, where h is the hamiltonian path angular frequency of $x(t)$ is needed to produce $y(t)$. Then for convenience angular frequencies may be scaled by multiplicative factor (just by changing time scale) to produce $f(t)$.

The purpose of the second part is to obtain n_h by effective low-pass filtering. When low-pass filtering is done naively by lowering cut-off frequency of the filter, this produces a long sequence of transient responses that cloud over steady-state response. In an ideal analog computation case, this transient response can simply be controlled by boosting to higher angular frequency. In many cases this is either not possible or undesirable, even though it is the main method presented in this paper. Thus a different mechanism is also proposed.

The idea is this: one can first filter for $1/2$ of upper angular frequencies of $f(t)$, double angular frequency then and then filter again. Repeat this procedure n^2 times, and one obtains the desired n_h. The main question then is how one doubles angular frequency. This is done by polynomial interpolation of each filter output, instead of relying on $f(t)$'s samples for every repeat of this procedure. Relying on filter output coming directly from $f(t)$ would require exponentially

Figure 2: The expanded walk representation of the graph in Figure 2.



many samples of $f(t)$ relative to $|V| = n$.

3 Low-pass Filter

Now that we defined the final output $k(t)$, the question is how we process $k(t)$ to give us some information about the number of hamiltonian paths, or C. To do this, we pass it to a low-pass filter. But we cannot simply assume an ideal low-pass filter, represented by $H(\omega) = rect(\omega)$, where $rect(\omega) = 1$ for $-0.5 < \omega < 0.5$ and $rect(\omega) = 0$ otherwise, because there is no such an ideal filter even to the approximate level.
Thus we will choose a simple physical first-order RC low-pass filter, described in figure 3.
By Kirchhoff's Voltage Law, the low-pass filter in figure 1 has the ODE of:

$$\frac{dV_{out}}{dt} + \frac{V_{out}}{\tau} = \frac{V_{in}}{\tau}$$

where $\tau = RC$. As this ODE is linear, to figure out the behavior of this low-pass filter, we first consider $V_in = De^{i\omega t}$, where ω is some arbitrary frequency.
Using initial capacitor voltage condition at the starting time $t = 0$ as $V_{out:t=0} = 0$,

$$V_{out} = \frac{D}{1 + i\omega\tau}\left[e^{i\omega t} - e^{-t/\tau}\right]$$

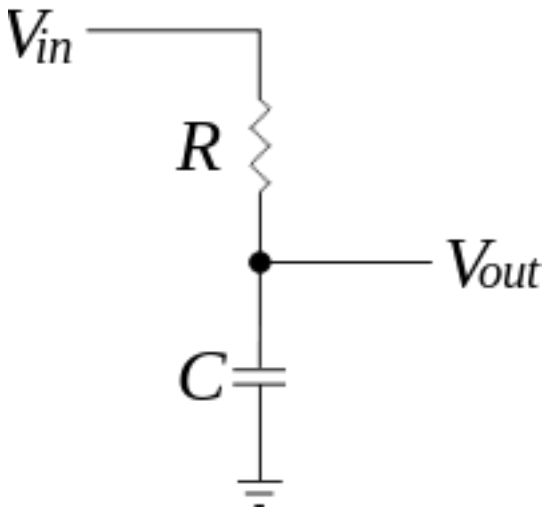


Figure 3: A first-order RC low-pass filter

Assume that $v_n > n + 1$. Also for calculation convenience, assume that $\tau = RC = 1$. In steady state $t = \infty$, because every ω of $j(t)$ except zero is greater than/equal to ${v_n}^{4n}$, and the total number of walks in G with n total vertices can only have maximum of n^n n-walks, $j(\infty)$'s value mostly comes from the hamiltonian/zero-frequency part. Other frequency parts only contribute less than $1/n^{3n}$ in magnitude. Thus at time ∞, the number of hamiltonian paths is discovered from the magnitude of $j(\infty)$, $|j(\infty)|$. However, calculations must be done on finite time, so the steady-state case only forms a background for our discussions, not the main part.
Note that in ordinary signal processing, keeping phase errors small is very important, but for the use of signal processing tools to analyze hamiltonian paths, phase errors are not of any concern.

3.1 Time Complexity of the Circuit

But moving to the finite time is simple: figure out the time when $e^{-t/\tau}$ decays to $1/n^{4n}$. Then high frequency parts only contribute a negligible value to $j(t)$. Now since $\tau = 1$ assumption is made, set equality $e^{-t_c} = 1/n^{4n}$. Taking the natural log to each side, $t_c = 4n \ln n < 4n^2$. Thus, the critical time, which is when the exponential decaying factor decays to $1/n^{4n}$, increases approximately linearly as the size of input n increases.
After this critical value, the value of $|j(t)|$ can simply be sampled by a digital computer to get the number of hamiltonian paths. Note that theoretically only one sample is required to measure the number of hamiltonian paths. This is because frequency 0 does not have any oscillating part, and thus will have

constant offset relative to 0.
Thus time complexity of the circuit to solve the number of hamiltonian paths is $O(n \log n)$, which is smaller than $O(n^2)$.

3.2 Size and Time Complexity

The above demonstrates that the number of needed components and needed time does not grow exponentially as the input graph size increase. All the values used in the circuit does not require exponentially-growing number of digits in a digital computer, as the graph size increases.

4 The Alternative Circuit Formation

In this section, I will describe another way of building a circuit that represents a graph. This method eliminates the use of frequency multipliers, and replaces them with ordinary multipliers.
Start with the original idea that each vertex x at column 1 transmits e^{ixt} to the wires x at column 1 are connected to. All wires going to vertex y at column $i > 1$ are first met with a sum operator, but now followed by an ordinary multiplier of $sum \times e^{iyt}$. The method will be explained in detail below.
Definition 1.1, 1.2, 1.3, 1.4 will be used as before. Section 1.1 changes to the following:

Definition 4.1 (The set V of vertex numbers)**.** The set V is defined as $V = \{n, n^2, ..., n^n\}$, which represents the set of vertex numbers (or equivalently vertex indices), with $|V| = n$, the number of vertices.

Definition 4.2 (n-walk)**.** A n-walk $\xi = (\xi_1, \xi_2, .., \xi_n)$ with $\xi_i \in V$ and $(\xi_i, \xi_{i+1}) \in E$ or $\xi_i = \xi_{i+1}$, a list, is a walk that has n vertices. A n-walk may contain self-loops or loops. One may consider a n-walk as a list of n vertex numbers that may contain one vertex number more than once.

Definition 4.3 (Permutation of a list)**.** A permutation of a list is a re-ordering of list elements of ξ.

Definition 4.4 (Uniqueness of n-walk frequency)**.** Let a n-walk ξ be $\xi = (\xi_1, \xi_2, .., \xi_n)$, which is a list. Let $\omega = \sum_{i=1}^{n} \xi_i$. ω is a unique n-walk frequency of G if it can only be the sum of some permutations of one list.

Lemma 4.1. *For $V = \{n, n^2, ..., n^n\}$, there cannot exist a n-walk frequency such that it is not unique.*

Proof. The proof is simply the basis representation theorem, except that the case where n vertex numbers that are same are in the list. In such a case, $\omega = n \cdot n^i$. But then $\omega = n^{i+1} = 1 \cdot n^{i+1}$, and $\xi = (n^{i+1})$ is the only possible alternative representation of ω. But the alternative list only has one vertex. Thus, there cannot exist a n-walk frequency that is not unique. □

Definition 1.5 needs to change as follows:

Definition 4.5. For each vertex x at column $a > 2$, if vertex y satisfies $(x, y) \in E$ or $(y, x) \in E$, vertex y mixer at column $a-1$, which multiplies e^{iyt} to a signal it receives, is connected by a wire to the sum operator at vertex x/column a. In case of each vertex x at column $a = 2$, if vertex y satisfies $(x, y) \in E$ or $(y, x) \in E$, vertex y oscillator (output of e^{iyt}) at column 1 is connected by a wire to the sum operator at vertex x/column 2.

Definition 1.6, 1.7 and 1.8 are kept. Definition 1.9 and 1.10 change to the following:

Definition 4.1. Each sum operator at vertex x/column $a \geq 2$ is connected to a mixer at the same column and the same vertex, which shifts frequency by x. A mixer, with shift factor of x, transforms $e^{iw_1 t} + e^{iw_2 t} + ...$ into $e^{i(w_1+x)t} + e^{i(w_2+x)t} + ...$, because it multiplies e^{ixt} to the signal it receives.

Definition 4.2. At column n, after signals pass through mixers connected to sum operators, any wire incident from column n is connected to a final sum operator instead, which produces the final signal $y(t)$.

Complexity remains the same: one needs $n(n-1)+1$ sum operators and $n(n-1)$ mixers/multipliers. (multipliers here are not frequency multipliers, but ordinary signal multipliers) The number of wires required remains the same.
Definition 1.11 changes to the following:

Definition 4.3. The final sum operator, which produces the signal $y(t)$ is connected to the ideal mixer M, which outputs the product of $y(t)$ with e^{-iut} where $u = v_1 + v_2 + v_3 + .. + v_n$, with $v_i \in V$. In Fourier transform, this is equivalent to converting $Y(\omega)$ with $Y(\omega + u)$, where $Y(\omega)$ is Fourier transform of $y(t)$. Let the output of M be $k(t)$.

Now $k(t)$ has zero frequency as its hamiltonian path frequency, as in the original formulation.
One may choose to add frequency multiplier after the final mixer M so that a simple first-order low-pass filter can be used. However, one may instead choose to increase the difference between each vertex number, such as $V = \{n, n^n, n^{2n}, ..., n^{n^2}\}$. This way, one does not have to add an extra frequency multiplier, which is likely to diverge from its ideal behavior, as I will discuss.

5 Real Deviations: Consideration of High Frequency and Frequency Multipliers

While the system described above is a physical system, not just a logical system, it is nevertheless still an ideal system. Oscillators are not perfect oscillators, resistors and capacitors are not ideal ones, wires have impedance. Thermal effects may change system properties.

But the most fundamental problem is the fact that the systems above are based on lumped-circuit analysis. Lumped circuit analysis works for low frequencies, because the length of wires can be made short enough to satisfy lumped-circuit assumptions. But one cannot shorten wires forever, and this makes lumped-circuit analysis to break for high frequencies. No longer discussion of lumped capacitors, resistors and inductors becomes a simple one.
At first, this seems to necessitate the need to discuss distributed circuits and transmission lines analysis. However, there is a recent technology that may allow us to think in terms of lumped circuit analysis.
The core idea behind the below method is time-stretching.

Step 1 Assume $V = \{n, n^2, .., n^n\}$, and the resulting $k(t)$. At Step 1, one first start with a low-pass filter of transfer function of $H(s) = 1/(s + 1/2)$. $H(s)$ has a cut-off frequency of $1/2$. And then one applies filter n^2 times, with a new filter operation starting after the previous filter operated for n^2 seconds. This results in time complexity of $O(n^4)$ seconds. Let the output be $k_1(t)$.

Note 1-2 Now assume hypothetically that the operating range of $H(s)$ is from 0 to $|\omega| = 2$, and for the frequencies inside the range, low-pass filtering works properly, but for other frequencies, it may be possible that some signals are not filtered. But these signals are not amplified.

Step 2 After Step 1, time-stretch the output $k_1(t)$ by factor of 2. That is the new time t' satisfies $t' = t/2$ for original t. Thus, angular frequency of 4 now becomes 2, and angular frequency of 2 now becomes 1. Angular frequency of 0 remains to be 0. The output is $k_1'(t)$. Repeat Step 1, but instead with input of $k_1'(t)$. The output is $k_2(t)$. Repeat the same process, which is time-stretching $k_i(t)$ by factor of 2, and low-pass filtering of $k_i'(t)$ and getting the output $k_{i+1}(t)$.

One continues the process until reaching $i = \log_2 n + 1$: by then, all angular frequencies are dealt with low-pass filtering. This allows the length of wires to be invariant even as n increases, and allows us to continue using lumped-circuit analysis. The above circuit process takes $O(n^5)$ seconds.
The question then now shifts to how time-stretching is done. One of the recent technology developed is photonic time-stretching, which is used for time-stretch analog-to-digital converters. The details of photonic time-stretching are wide-ranging, and thus I will not discuss these details. However, the only three requirements for practical time-stretching imposed by the paper's method are: 1. DC signal inside $k(t)$ is kept as close as possible, 2. frequencies close to the range from $\omega = -1$ to $\omega = 1$ are kept mostly zero when $V = \{n, n^2, .., n^n\}$, 3. some deviations from ideal filtering behavior for $|\omega| > 1$ are fine only if they do not significantly change amplitude behaviors. [Bhushan 1998], as one of first examples of applying photonic time-stretching, can be referenced for more information.
The above implicity assumed optical-electrical signal converter and vice versa,

which may be ideal or non-ideal. This part would not be discussed.
Now into less serious problems. If errors introduced by real deviations affect the zero-frequency part below a certain threshold, they may safely be ignored. For example, multiplying by e^{-iut} may not shift frequency u to 0 in real-time systems. Usually, however, frequencies do spread out, and often 0 frequency part is not emptied.
While there are many non-ideal issues that affect oscillators (and possibly for mixers and frequency multipliers too), if we assume time-decaying realistic oscillators, Q-factor may be used to gauge this performance part. Assume that this decaying time is associated with our measurement time also - which means that measurement time is just enough to allow us computation before signals almost disappear completely. More theoretically, Gabor limit is there:

$$\sigma_t \sigma_f \geq \frac{1}{4\pi}$$

While the above formula only gives the bound, assume that every system has equal $\sigma_t \sigma_f$. If our measurement time increases, so must decaying times. Representing this as increase in σ_t, σ_f will decrease. In case of an oscillator, this is equal to becoming close to an ideal oscillator. Thus increasing necessary decaying time will help the performance of oscillators. As thus, the size of input is not an extra constraint for Q factor problems of real-time systems.
Many problems, whether small or not, require more details are left out here. Future papers will address these issues.

6 Digitization

Here, I will follow the alternative circuit formation. I will re-formulate the circuit formulation as the grid formulation as follows. Thus, for digitization, while the above sections serve as inspirations and direct connections can be found, this section itself does not directly reference other sections. All notations in this section are separate from other sections.

Definition 6.1 (Z^+, Z_+, Z^-, Z_-)**.** Z^+ or Z_+ refers to the set of positive integers. Similarly, Z^- refers to the set of negative integers.

Definition 6.2 ("less than", "more than", "greater than", "smaller than")**.** Unless otherwise noted, these are all comparisons in magnitude/size/absolute value.

Definition 6.3 (Base-n expansion)**.** Base-n expansion of some number k is basically expressing k in base-n: $k = \pm \sum_{p=-\infty}^{\infty} a_p n^p$ with $0 \leq a_p < n$.

The power of base-n is that if important parameters are the finite sums (that is, $k = \pm \sum_{p=b_l}^{b_h} a_p n^p$, with b_l and b_h finite), instead of infinite sums, then analysis becomes much easier. For studying numerical approximation of k (if exact value cannot be known), one can just focus on finite number of numerical digits.

Definition 6.4 (graph, n)**.** A graph G is denoted with $G = (V, E)$ as done in the standard literature. $n = |V|$ is assumed whenever n appears.

Definition 6.5 (walk, n-walk, hamiltonian path)**.** A walk is defined as in the standard graph theory vocabulary. A walk that has n vertices is called n-walk. Let us represent a walk with a list (tuple) of vertices in a traversing order from the start vertex to the end vertex. By the definition of a walk, one vertex can appear more than once in a list. A hamiltonian path, as defined in the standard graph theory vocabulary, is a walk with n distinct vertices, where $|V| = n$.

Definition 6.6 (vertex)**.** A vertex is assigned a number. Each distinct vertex has a distinct number. Let $V = \{n, n^2, n^3, ..., n^n\}$. From now on, one can assume a vertex as a number whenever appropriate.

Definition 6.7 (n_h, n_p)**.** n_h is the number of hamiltonian paths of G. n_p is the total number of n-walks of G.

Definition 6.8 (Vertex-number)**.** The vertex-number of a walk is defined as the sum of all elements (vertices) in the list of a walk.

Note that the vertex-number of a walk represents the angular frequency of a walk in $x(t)$, as will be seen. It is certainly possible that two walks may occupy the same frequency. If there are k walks that occupy the same frequency ω_a, then the amplitude at the frequency would be k in Fourier series language, or $k\delta(\omega - \omega_a)$ in Fourier transform language where $\delta(\omega)$ is a dirac delta function. **The maximum number of vertices inside a walk is restricted to n, for sake of convenience.**

Definition 6.9 (Permutation of a list)**.** A permutation of a list is a re-ordering of list elements in ξ. For example, for $\xi = (\xi_1, \xi_2, .., \xi_n)$, $\xi_\alpha = (\xi_n, \xi_{n-3}, \xi_{n-4}, ..., \xi_1)$ is a permutation of ξ.

Lemma 6.1. *Given V as defined above, a vertex-number can only be formed out of a permutation of a single vertex-number list.*

Proof. The proof is simply the basis representation theorem, where basis are elements in V. One exception to this proof, though, arises when a list ξ representing a walk may be of $(k, k, ...k)$ with $|\xi| = n$ and $k = n^i$, or in words, there are n k's in ξ. In this case, $nk = n^{i+1}$, meaning the vertex-number ξ equals one of vertices in V. But this should not matter whenever walks one deals with have same number of vertices. □

Following from above:

Definition 6.10 (Contribution of each n-walk to $x(t)$)**.** From above, each walk has a vertex number k. Each n-walk is said to contribute e^{ikt} to $x(t)$.

Definition 6.11 (Amplitude)**.** For any arbitrary function $\alpha(t)$ expressible as $\alpha(t) = \sum_{\omega=-\infty}^{\infty} A_\omega e^{i\omega t/d}$ where d is constant and does not vary with ω, A_ω is said to be amplitude of $\alpha(t)$ at angular frequency ω.

6.1 Grid: $x(t)$

Definition 6.12 (Grid, wires)**.** A grid consists of n depths, with each depth being equivalent to a column. Each depth contains n vertices as in V. Each wire connects a vertex v_α from ith depth to a vertex point of v_β in $i+1$th depth. A wire is connected between v_α to v_β if and only if $(v_\alpha, v_\beta) \in E$.

Definition 6.13 (Function transmission: first depth case)**.** In the first depth (first column), each vertex v_α transmits $e^{iv_\alpha t}$.

Definition 6.14 (Function transmission except for first and nth depth)**.** Defining for each v_α in arbitrary ith depth. All incoming wire transmissions $w_\zeta(t)$ from each wire ζ from $i-1$th depth to v_α in ith depth are summed, or equivalently $w_\lambda = \sum_\zeta w_\zeta$. And then multiply by $e^{iv_\alpha t}$ and transmit $u_{v_\alpha} = e^{iv_\alpha t} w_\lambda$ to each wire starting from v_α.

Definition 6.15 (Vertex point function transmission: nth depth case)**.** All incoming wire transmissions $w_\zeta(t)$ from each wire ζ from $n-1$th depth to v_α in nth depth are summed, or equivalently $w_\lambda = \sum_\zeta w_\zeta$. And then multiply by $e^{iv_\alpha t}$, resulting in $s_{v_\alpha} = e^{iv_\alpha t} w_\lambda$. $x_{ideal}(t) = \sum_{v \in V} s_v$ is the output of the grid, not considering quantization errors involved.

For each depth i, $\sum_{v \in V} u_v$ shows the sum of all vertex-numbers representing i-walk.

6.2 Post-grid: $y(t)$

Simply, this post-grid procedure is all about calculating $y(t) = x(t)e^{-iht}$ where $h = \sum_{i=1}^{n} n^i$, the hamiltonian frequency of $x(t)$. Thus, $y(t)$ has 0 has hamiltonian frequency.

6.3 Post-grid: $f(t)$

$f(t)$ is defined as $f(t) = y(ct)$. c will be defined later as $c = 1/n^{n+1}$, but the choice of c is irrelevant except matter of convenience.
Let the angular frequencies of $f(t)$ be labelled with u. $u = 0$ refers to hamiltonian frequency.
From now on, when it is said "every u," this refers to every u with non-zero amplitude in $f(t)$.

6.4 Sinusoidal quantization errors

For every vertex v of each depth of the grid, the numbers from maximum of n vertices are added and then multiplied by e^{ivt}, for each t.
For each vertex v of each depth, the error occurred would be of the following form:

- (Sum of errors from previous depths) × (e^{ivt} + calculation error for e^{ivt}) + (The correct sum of previous depths) × (calculation error for e^{ivt}).
- The item sign may be misunderstood as a negative sign, but the first bar in the first item refers to an item sign.

Here the purpose of error analysis is not to find out exact error but to derive the formula for the magnitude that is equal or bigger than actual possible maximum error.
Assume that the correct value of the previous depth is always $2n^n > |n^n + n^n i|$, and the correct value of e^{ivt} is $2 > |1 + 1i|$. This is bigger than it actually is, thus Equation 1 is an overestimate of the sum of errors.

Definition 6.16 (e_v, e_i)**.** e_i representing total maximum sinusoidal quantization error in magnitude occurring from depth 1 to depth i of all vertices, and e_v represents the maximum error in magnitude that occurs from calculating e^{ivt}.

Note that e_v and e_i represents entirely different things, and v inside e_v is not an index, unlike i, which is an index, in e_i.
Thus, this will yield the following recurrence equation:

$$e_{i+1} = n^2 \left[(2 + e_v)e_i + 2n^n e_v\right] \tag{1}$$

Now let us simplify Equation 1 by the following substitutions:

$$\Lambda = 2n^2 + n^2 e_v, \ \Upsilon = 2n^n e_v \tag{2}$$

$$e_{i+1} = \Lambda e_i + \Upsilon \tag{3}$$

Assuming that we start from $e_0 = 0$ (for sure, depth 0 does not exist, but this can safely be used), by geometric series formula,

$$e_i = \Upsilon \frac{\Lambda^i - 1}{\Lambda - 1} \tag{4}$$

$$e_{i=n} \equiv e_n = \Upsilon \frac{\Lambda^n - 1}{\Lambda - 1} < \Upsilon \Lambda^n \tag{5}$$

Assuming that $e_v < 1/n^2$, we can assume that $\Lambda < 3n^2$.
To incorporate the errors occurring from further calculating $y(t)$,

$$e_{i=n+1} \equiv e_{n+1} = \Upsilon \frac{\Lambda^{n+1} - 1}{\Lambda - 1} < \Upsilon \Lambda^{n+1} \tag{6}$$

with $\Lambda^{n+1} \approx 3^{n+1} n^{2n+2} \approx n^{3n+3} \approx n^{4n}$, assuming $e_v < 1/n^2$.

6.5 n_h extraction: interpolation-filtering cycle

n_h extraction is simply digitization of low-pass filtering. However, naive digitization faces the following problem:

- The cut-off frequency is $1/n^{n+1}$ of maximum frequency of $f(t)$. Naive digitization results in transient response noise clouding over steady-state response for exponential amount of time, thus rendering the procedure meaningless.
- In general, trade-off exists between better low-pass filtering (by reducing cut-off frequency) and transient response (by increasing cut-off frequency, which moves s-pole in Laplace transform terms to the left), and optimal ground often does not exist.

To avoid this problem, interpolation-filtering cycling mechanism is proposed in this subsection.
Let $i(t)$ be "generic" input to the filter, with maximum frequency with non-zero amplitude at $\omega = 1$. In this mechanism, a Butterworth filter:

$$H(s) = \frac{0.5}{s + 0.5} \tag{7}$$

will be used several times, with $i_a(t)$, input to the ath use of the filter, being either interpolation result of $k_{a-1}(t)$ with frequency doubled and phase changed or simply filter output $k_{a-1}(t)$ corresponding to input $i_{a-1}(t)$, depending on a. This butterworth filter can be converted to digital domain by bilinear transform with sampling interval T. T will mainly be adjusted to get desired precision level for interpolation. I will set $T = 1/n^{16}$.
While interpolation problem will be discussed in details, I will first present what interpolation has to be done.

- The given filter output has maximum angular frequency with non-zero amplitude at $\omega = 1/2$. (Or one can treat as if this is the case, for after $\omega = 1/2$, amplitude is so small as to be ignore-able as part of quantization errors.)
- One has to interpolate n^{20} samples of the given filter output function (that one only has samples) with equal spacing $2T$, or $\Delta t = 2n^4 = 2n^{20}T$ in time interval of these samples, from the given endpoint sample (that one already has) at t_e of one cycle. The fact that spacing is $2T$ implies this is taking the original filter output and doubling its frequency with phase distorted. (thus maximum angular frequency can now be treated as 1 for interpolated samples - see the definition for steps and cycles below.)
- This is done using n^{16} samples in the unit interval $\Delta t = 1$ before t_e that one already has.
- By polynomial interpolation error formula, upper bound on interpolation error magnitude is less than $1/n^{4n^{16}-4n^8} \approx 1/n^{n^{16}}$ for each interpolated sample. (Recall the first item/bullet point.)

While equal spacing interpolation is used in this paper, for numerical improvement, other spacings may be used with minor adjustments to interpolation

procedures.
Now one has it: for each sample, $O(n^2)$-digit precision is used, there are n^20 samples that needs to be dealt for interpolation. Interpolation calculation sensitivity to numerical error is already considered as part of error-controlling - and one can treat these errors as adding up to errors in zero frequency amplitude. (the upper bound of error for each filter output/input sample can be considered as the upper bound of error for calculating n_h. And one knows that n_h must be a non-negative integer.) As shown above, this is so small as to be ignore-able.
For computational complexity, one can take sufficient measure of $O(n^{498})$ for each interpolation problem. While actual time complexity will be much smaller than this obviously, for saving space I will not go into calculating actual time complexity. As long as it is known that complexity is not exponential to n, the paper serves its intention.
There will be n^2 interpolations - meaning that sufficient time complxity is $O(n^{500})$ - this dominates other time complexity considerations, and thus total procedure amounts to $O(n^{500})$.
Now to explanations of why the interpolation problem as posed is needed.

Definition 6.17 (steps,cycles)**.** Steps are part of a cycle. For each step a in cycle b, there is filter input $i_{a,b}(t)$ with maximum angular frequency with non-zero amplitude (or treated effectively as so) at 1. After taking $\Delta t = n^2$ (n^{18} samples), transient response decays to sufficient intended low-pass filtering purpose of the first-order Butterworth filter to produce $o_{a,b}(t)$. But one wishes to obtain much more low-pass filtering, and one filters n^2 times - implying total of n^2 steps for each cycle. At the last step of each cycle, interpolation is done as to double the frequency with phase distorted. This is done by taking the last n^{16} samples and interpolating next n^{20} samples, as shown in the above bullet points for interpolation.

The followings are the important points for interpolation/steps/cycles:

- Phase does not matter for finding n_h, but for convenience one may keep phase at zero frequency as zero (and this is indeed done by filter laplace transform equation in 7). Phase at other frequencies do not matter. Thus phase distortion does not matter.
- Filter gain response at zero frequency is 1.
- Effectively, moving from one cycle to the next cycle moves down cutoff frequency from the perspective of the first filter input $f(t)$, but without trade-off between cutoff frequency and transient response control.

Interpolation is the crucial part of this digitization scheme, for:

- Naive approach of reducing down cut-off frequency makes transient response control very difficult.
- Naive approach of not interpolating but using original samples of $f(t)$ to simulate doubling frequency does not work, as this approach requires exponential number of samples of $f(t)$ relative to n.

Now let us see how much transient response decays for each step and why the number of cycles and steps are justified. Since one uses $\Delta t = n^2$ for each step to wait for decaying steady-state response, decaying factor is $e^{-n^2/2}$ of its original transient response. At maximum, initial transient response has magnitude of n^n, and one can see this decaying factor is enough to make remaining transient response to be irrelevant.
n^2 steps for each cycle is to ensure that at $\omega = 1/2$ of the cycle filter input, steady-state response guarantees that magnitude of amplitude decays by $1/2^{n^2/2}$. This makes steady-state response from angular frequency 1/2 to 1 ignore-able. Thus, when frequency is doubled by interpolation mechanism (and passing appropriate input to a new cycle), these frequencies from 1/2 to 1 (or 1 to 2 in new cycle input) can be ignored.
Number of cycles is n^2, because one wishes to reach cut-off frequency $1/n^{n+1}$ in terms of original input. Because one doubles frequency after each cycle, $1/2^{n^2} < 1/n^{n+1}$ is sufficient to reach the desired cut-off frequency.
To summarize, one needs n^18 samples for each step with sampling interval $T = 1$, n^2 steps for each cycle, and there are n^2 cycles. At the end of the cycle, interpolation is done to double frequency with phase distortion.
Thus,

$$y(t/n^{n+1}) = f(t) \tag{8}$$

for the grid mechanism to ensure that maximum angular frequency is 1. The whole digitization scheme amounts to $O(n^{500})$ time complexity.

7 Conclusion

This paper introduces an analog circuit, involving a $n \times n$ grid (subcircuit) and a low-pass filter that allows us to compute the number of hamiltonian paths in an ideal physical environment, with the assumption of ideal oscialltors/mixers/frequency multipliers, with a different degree of relaxation also examined. Then the paper formulates time complexity of such a circuit and concludes that it is $O(n \log n)$, with non-exponential space complexity. Also for a certain non-ideal case described in this paper, time complexity is $O(n^5)$. However, in most cases, analog computation is undesirable, and a possible digitization scheme is proposed.

8 References

Baumgardner, J. et al. (2009). 'Solving a Hamiltonian Path Problem with a bacterial computer', Journal of Biological Engineering, 3(11): 109–124.

Bhushan, A.S. et al. (1998). 'Time-stretched analogue-to-digital conversion', Electronic Letters, 34(9): 839–841.

Haist, T. et al. (2007). 'An Optical Solution For The Traveling Salesman Problem'. Optics Express, 15(16): 10473–10482.

Rosser, J. (1941) 'Explicit bounds for some functions of prime numbers', Amer. J. Math. 63, 211-232. MR 2:150e

Sartakhti, J. et al. (2013). 'A new light-based solution to the Hamiltonian path problem'. Future Generation Computer Systems, 29(2): 520-527.