Quantum Walks with Time-Dependent Hamiltonian

Matteo Garbellini
Department of Physics
Università degli Studi di Milano

(Dated: July 23, 2020)

I testi in rosso sono commenti su argomenti mancanti, informazioni da aggiungere e analisi necessarie per completare il lavoro. Una lista piu o meno completa si trova nella Sezione V

I. INTRODUCTION

This report can be summarized as follows.

- In Section II a review of the background topics is presented, with particular enphasis on Quantum Walks, Quantum Search and the Adiabatic Theorem. A brief introduction on Graph Theory is presented as well, in order to define the working context.
- In Section III we discuss the Dynamics, focusing on the time dependent and time independent hamiltonian approach to quantum search. A definition of the computational routines used throughout the work is also given.
- In Section IV we present results for selected topologies, namely the *worst-case-scenario* Circular Graph and the *best-case-scenario* Complete Graph. For the two selected topologies we compare the known time-independent approach and the time-dependent, in particular defining two qualitative characteristics for results: optimized search and localization.
- Section V is a list of things yet to be done/finished.
- In the Appendix a discussion on the computational methods, in particular the differential equation solver and the optimization algorithm used.

II. BACKGROUND

A. Graph Theory

A graph G is defined as a ordered pair (V, E), where V is a set of vertices and E is a set of edges, which represent the connection between any two pair of vertices. If indeed any two vertices (i, j) are connected by an edge we define as adjacent, and from this we can construct the *adjacency matrix* A as:

$$A_{ij} = \begin{cases} 1 & (i,j) \in G \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

which represents the connectivity of the graph. We can then describe the degree of each vertex of the graph, namely the number of vertices (excluding itself) connected to it, through the *degree diagonal matrix* $D_{jj} = deg(j)$.
It is also useful to introduce the *laplacian matrix* defined as

$$L = A - D \tag{2}$$

which we'll later see describes the quantum walk evolution.
In the quantum realm a vertex is a vector in an N dimensional Hilber space, so that a vertex $|j\rangle$ can be represented in the following way

$$|j\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ .. \\ 0 \end{pmatrix} \tag{3}$$

Throughout this work we focus our attention on two particular graph topologies: the **circular** graph and the **complete** graph (FIG. 1).

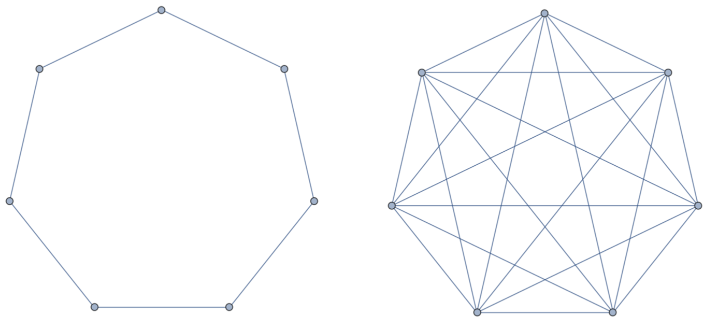


FIG. 1. Pictorial representation of a circular graph (left) and a complete graph (right) with N=7

B. Quantum Walks

The continous time quantum walk is the direct analougue of the classical random walk. Given a graph G, a random walk is a Markov process with a fixed probability for unit time γ of jumping to an adjacent vertex j. This particular walk can be described by a linear differential equation in terms of probability, namely

$$\frac{d}{dt}p_j(t) = \gamma \sum_k L_{jk} p_k(t) \tag{4}$$

where $p_j(t)$ is the probability of being on the vertex j at time t.
The quantum analogue takes place in an N-dimentional Hilbert space spanned by the states (vertex) $|j\rangle$ of the graph G. Instead of considering the classical probability as we previosuly did, we consider the probability time-dependent amplitudes $q_j(t) = \langle j|\psi(t)\rangle$, where $|\psi\rangle$ is a general time-dependent state. The differential equation takes thus the form of

$$i\frac{d}{dt}q_j(t) = \sum_k H_{jk} q_k(t) \tag{5}$$

The continous-time quantum walk is defined by letting $H = -\gamma L$, where L is the previosuly defined Laplacian matrix.

C. Grover's Quantum Search

This section should include the standard Grover's Quantum Search to give the contex for the quantum walk approach.

D. Quantum Search with Continous-Time Quantum Walk

***Spatial Search by Quantum Walk*, A. Childs, J. Goldstone, quant-ph/0306054v2**
We now address the quantum search problem firstly formulated as Grover's algorithm and then extending it to the search on a graph using quantum walks.
To approach the Grover problem with quantum walk it's necessary to modify the hamiltonian such that the vertex $|w\rangle$, i.e. the target, is somewhat special. Following Grover's oracle an oracle hamiltonian H_w is introduced

$$H_w = -|w\rangle\langle w| \tag{6}$$

which in particular has energy zero for all but the vertex $|w\rangle$ for which it has enenergy -1. Therefore the Grover problem, i.e. quantum search, becomes finding the ground state of such hamiltonian. To do so we consider the time-independent hamiltonian of the form

$$H = -\gamma L + H_w = -\gamma L - |w\rangle\langle w| \tag{7}$$

where L is the laplacian of the graph, which contains the information of the dynamics over that particular graph topology. The evolution of the quantum walk is therefore governed by this hamiltonian.

The quantum search routine works as follow:

- we consider the superposition of all possible states, namely

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_j |j\rangle \tag{8}$$

- we find the evolved state using the hamiltonian for a time T H

$$|\psi(T)\rangle = U(T)|s\rangle = \exp\Big\{ -\frac{i}{\hbar} HT \Big\} |s\rangle \tag{9}$$

  (Note that this evolution is valid only for time-independent hamiltonians.)

- we then measure the state onto the target $|w\rangle$ and find the corrisponding probability

$$p = |\langle w|\psi(T)\rangle|^2 \tag{10}$$

The objective is to find the optimal value of γ so that the probability of the system of finding itself in $|w\rangle$ is as close as possible to 1 for the smallest T.

E. Search on Complete Graph

We now look at the search on a complete graph. This case is particularly interesting since it can be solved analitically to be continued

F. Adiabatic Theorem

***Quantum Computation by Adiabatic Evolution*, E. Farhi, J. Goldstone, S. Gutmann, M. Sipser, quant-ph/0001106**

A quantum system evolves according to the Schroedinger equation

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle \tag{11}$$

and defining the instantaneous eigenstates and eigenvalues of H(t) by

$$H(t)|l;t\rangle = E_l(t)|l;t\rangle \tag{12}$$

such that $E_0(t) \geq E_1(t) \geq ... \geq E_{N-1}(t)$.
The adiabatic theorem states that if the gap between the two lowest energy levels, $E_1(t) - E_0(t) > 0$, is strictly greater than zero then for $T \to \infty$ the probability of being in the ground state is equal to one, namely

$$\lim_{T\to\infty} |\langle l = 0; t = T|\psi(T)\rangle| = 1 \tag{13}$$

This means that if the system is chosen to evolve at a slow enough rate, the instantaneous hamiltonian will remain in the ground state throught the evolution. It is useful to consider a smooth one-parameter hamiltonian $H(s)$ such that $s = t/T$, with $t \in [0, T]$ so that $s \in [0, 1]$. Let's now define the energy minimum gap by

$$g_{min} = \min_{0 \leq s \leq 1} (E_1(s) - E_0(s)) \tag{14}$$

In addition we can find a time lower bound T^* such that for $T \gg T^*$ the probability is arbitrarily close to 1, in detail

$$T \gg \frac{\varepsilon}{g_{min}^2} \tag{15}$$

where

$$\varepsilon = \max_{0 \leq s \leq 1} \left| \left\langle l = 1; s \middle| \frac{dH(s)}{dt} \middle| l = 0; s \right\rangle \right| \tag{16}$$

Let's now discuss how to take advantage of the adiabatic theorem introducting the usual way in which the adiabatic evolution is implemented. It is often presented a problem hamiltonian H_P whose ground state is not so straight forward to find; on the other hand we can prepare the system in abeginning hamiltonian H_B whose ground state is known. The problem hamiltonian encodes the solution of the problem, while the beginning hamiltonian is a tool for easily preparing the state to be evolved. The adiabatic implementation then consists, assuming that the ground state of H_P is unique, in having a time dependent hamiltonian $H(s)$ such that

$$H(s) = (1 - s)H_B + sH_P \tag{17}$$

In this way we can prepare for $s = 0$ the system in H_B and let it evolve so that for $s = 1$ it reaches H_P. Thanks to the adiabatic theorem, if it's made to evolve sufficiently slowly we will find ourself in the ground state of the problem hamiltonian, which is exactly the solution.

III. DYNAMICS AND METHODS

In this section we look more in depth at the time-independent and time-dependent hamiltonian used for the search algorithm, putting particular focus on the time function that regulates the evolution of the hamiltonian. We then explore the possible way to numerically solve the schroedinger equation, giving reasoning of why we decided to solve it numerically using a Runge-Kutta integrator. Last but not least we look at the computational methods, namely the code written and the optimization and integrator algorithms used.

A. Time Independent Hamiltonian

As we've seen in Section ?, the quantum search with quantum walk consists in finding the probability $p = |\langle w|\psi(T)\rangle|^2$, thus we're interested in the evolution of the beginnig state $|s\rangle$ using the hamiltonian:

$$H = -L - \gamma |w\rangle\langle w| \tag{18}$$

that differs from the hamiltonian introduced by Goldstone & Childs having the γ term in front of the oracle instead of the laplacian. The two, however, describe the same evolution.
The evolution of a state for the time independent hamiltonian is given by the operator U(t)

$$U(t) = e^{-\frac{i}{\hbar}Ht} \tag{19}$$

Therefore the probability is given by the following

$$\begin{aligned} p &= |\langle w|\psi(t)\rangle|^2 \\ &= |\langle w|U(t)|s\rangle|^2 \\ &= |\langle w|e^{-\frac{i}{\hbar}Ht}|s\rangle|^2 \end{aligned}$$

From a computational point of view this evolution is a simple matrix multiplication, and doesn not require any particular tool.

B. Time Dependent Hamiltonian

We introduce a time dependent hamiltonian that follows from the adiabatic implementation discussed in Section ? and the time-independent approach previously discussed. The intuition behind this implementation comes from the idea of *adiabatically* turn on the oracle hamiltonian. Theoretically this should give us the advantage discussed in (Section II - Adiabatic Theorem), namely the high probability of finding the solution with the possible expense of longer time scaling.
The hamiltonian is thus in the form of

$$H(t) = (1 - s(t))L + \gamma s(t)|w\rangle\langle w| \tag{20}$$

where $s(t)$ is a function of time that will be later discussed in more detail.

The probability is calculated similarly to the time-independent hamiltonian, with the exception that the time evolution operator is given by

$$U(t) = \exp\Big\{ - \frac{i}{\hbar} \int_0^t dt' H(t') \Big\} \tag{21}$$

This makes finding the evolved state $|\psi(t)\rangle$ not so straight-forward, in particular for the circular graph for which the hamiltonian does not show any particular pattern/property. A few approximation could be used for this task such as the *Dyson Series* and the *Magnus Expansion*. However, for our specific scenario, for which we only needed the evolved state we decided to opt for *solving the Schroedinger equation*, namely

$$i\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle \tag{22}$$

Since H is a matrix we had to solve it for components. Recalling that $|\psi\rangle$ is a vector in an N-dimensional Hilbert space, we need to solve N-differential equations of the form:

$$\frac{d}{dt}|\psi_i(t)\rangle = \sum_j H_{ij}|\psi_i(t)\rangle \tag{23}$$

Once the N solutions are found, the probability is easily found with $p = |\langle w|\psi(t)\rangle|^2$. It's worth noting that some additional normalization on the found solution might be necessary due to the integrator used, however a more in depth discussion on the integrator and normalization can be found in the Appendix.

C. Comments on the form of s(t)

Discussion on the time-stepping function $s(t)$ (or better $g_T(t)$). Why did we choose this particular shape for the function? Are we expecting any improvements?
In particular if we follow what has been said by Cerf et al. the function should be steeper (evolving faster) when the separation $E_1(t) - E_0(t)$ is larger, and slower for small separation. Looking at the separation distribution the functions used (sqrt, cbrt, linear) do not fit to this model.
It's interesting to show, probably in the Results section rathen than here, how significant is the shape of s(t) in the final probability

D. Computational Routines

Throughout the analysis the goal was to find the maximum probability for the specific scenario, and the procedure varied depending on the graph topology and the hamiltonian considered. We can therefore characterize the three main computational routines used:

- **γ-T optimization:** as it might imply, we run an optimization on the probability using γ and time as parameters, trying to maximize the probability. This is used for the time-independent search, since it's computationally inexpensive.

- **T optimization for fixed γ:** we sample a fix number of γ values and optimize on the time variable. This is useful to gain insight on the qualitative probability distribution for varying time.
- **probability grid evaluation:** this technique consists in sampling both time and γ and evaluating the probability for all the (γ, T) combinations. The advantage of such approach is the much less computational time needed since a single evaluation is much faster than an optimization over the (γ, T) space. Secondly true optimization tries to find the absolute maximum, to the expence of having a larger T, while we're interested in a probability trend rather than a very precise maximum.
  As an example, we might find a probability p=0.95 for T = 10 using the grid evaluation, although the true maximum is p=0.96 for T = 35. It iss indeed clear that the 0.01 in greater probability does not balances the disadvatange of more than double time. this is made up data, might be interesting to show *actual* data.

Additional information on the optimization algorithms used can be found in the Appendix.

IV. RESULTS FOR SELECTED TOPOLOGIES

In this section we first characterize two classes of results, namely the optimized search and the localization. We then look at search results with the time-independent hamiltonian and compare them to the time-dependent approach. In particular we look at the *best-case-scenario* the **complete graph** for which we know the analitical solution and the *worst-case-scenario* the **circular graph** for which no analytical solution is yet known.

A. Search vs Localization

We shall describe the possible outcomes of the quantum search and characterize them as **localization** and **optimized search**:

- The first describes the finding with high probability of the solution without particular interest in the time needed to reach that particular solution. As we will later see this is a direct consequence of the adiabatic theorem.
- The latter describes the finding of the solution with high probability, optimizing for the time as well. In addition as we shall see later we'll also consider the possibility of repeating the search an n amount of times, and get a less than optimal search for each iteration but better time (sum of time, actually) overall.

B. Circular graph

1. *Time-Independent Benchmarks*

The first step in our analysis is to compute some benchmark for the time-independent hamiltonian, in order to later compare the time-dependent approach. This is particularly interesting since we know that th quantum search time-independent algorithm does not work with the circular graph, namely the found probability decreases with N increasing.
Using the time-independent hamiltonian introduced before we optimized the probability on T and β for circular graphs up to N=31. It is worth noting that we considered only odd graphs, since it made an easier center-placement of the oracle site state. The following are the results:
figure: time-independent benchmarks up to N=31

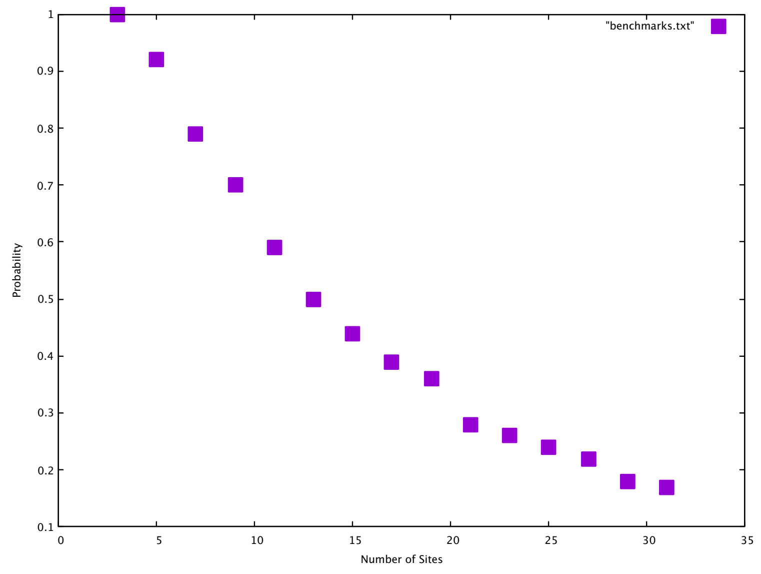


FIG. 2. Time Independent Benchmarks up to N=31

It is clear that the time-independent algorithm does indeed not work, as we expected. The probability decreases as the number of sites N increases, I noticed that for large N the probability (fixing γ) converges to $p = 0.1$. Further investigation is needed.

2. *Time-Dependent Algorithm*

We then considered the time-dependent hamiltonian introduced in Section ?, and following the *probability grid evaluation* procedure discussed before we computed the probability. Given the computational efficency over the regular β-T optimization we were able to compute results up to N = 71, although from 31 to 71 only the 1st and 7th (e.g. 41,47) were calculated.

It is however clear that the grid-evaluation approach does not produce unique values of (T, β) corrisponding to the absolute probability maximum. On the other hand is able to give a qualitative idea on the probability distribution for various (T, β) combinations. We then present the found data with a **heatmap plot** such the one below, which makes quite easy to visualize the result.

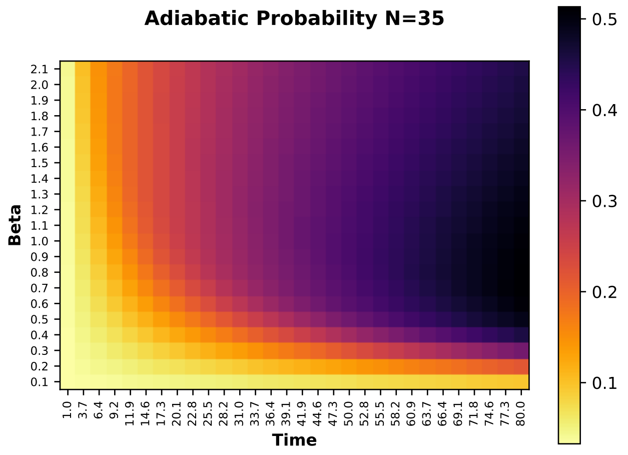


FIG. 3. Time Dependent heatmap example, N=35

The task now is two develop a method to compare the time-independent and time-dependent algorithms. We first naively look at a qualitative comparison that immediately shows that the time-dependent algorithm is able to find higher probabilities but at the expense of worst time scaling; secondly we introduce a quantity to compare the two methods numerically.

3. *Qualitative Comparison*

In order to qualitatively compare the two approaches to quantum search we will consider a heatmap plot where we superimpose probability **contour lines** corrisponding to the following probability (and with the following format):

- (solid) p = 0.99, p = 0.95, p = 0.90 for the time-dependent results
- (dashed) the probability found in the corrisponding time-independent results

These, with the corrisponding time T at which the time-independent probability is found (given by a vertical line), should give a qualitative method to compare the two approaches. How shall we do so?

- Look for any intersection between the dashed contour line and the vertical line
- If it does intersect the intersection point(s) represent a (γ, T) combination that is equivalent as the time-independent algorithm in terms of probability

- If the dashed contour line and the vertical line identify a bounded region (might be necessary a figure to explain this), that particular region represents a (γ, T)-space for which the time-dependent algorithm performs better than the time-independent one.

4. ***Numerical Comparison***

In order to numerically compare the two approaches we define a quantity δ in the following way:

$$\delta = \min\left(\frac{T}{p}\right) \tag{24}$$

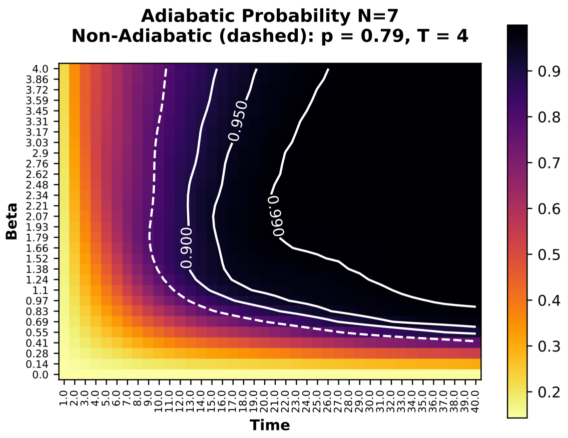
The quantity T/p represents the total time necessary to get to unitary probability. Finding the minimum of that quantity corresponds to finding the best combination of T and p that gets to unity with the minimum T.

What we do is thus for each vertical column of the grid, i.e. constant T and variable β, computing the maximum probability and finding the quantity T/p. We than find the minimum of all the T/p calculated. In this way we **reduce a grid to a single value** to compare to the time-independent results.

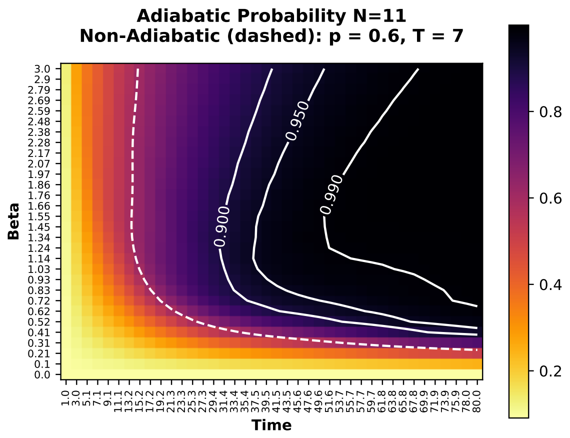
The following plot compares the time-independent approach to the time-dependent one, using for the first the quantity T/p evaluated with the β-T *optimized* probability while the latter uses the quantity δ.

5. ***Different s(t) shapes***

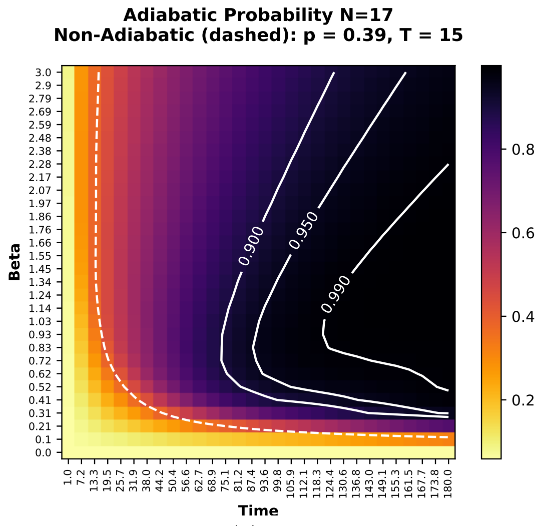
As we introduced in Section ? we're interested in determining how influencial is the shape of s(t) in the probability distribution. We then proceed as in Section ? using the step function s(t) previously discussed.
To compare we consider the quantity δ. Since we're interested in the trend for large N we ignore small N and only consider a few sampled N between 35-70.



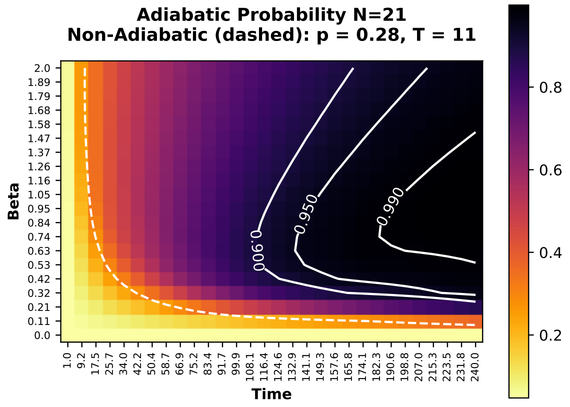
(a) N = 7



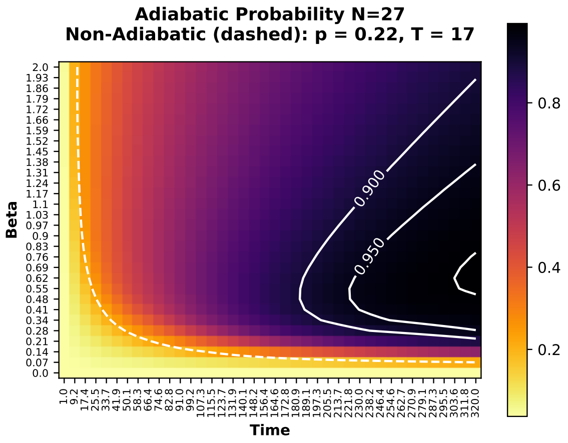
(b) N = 11



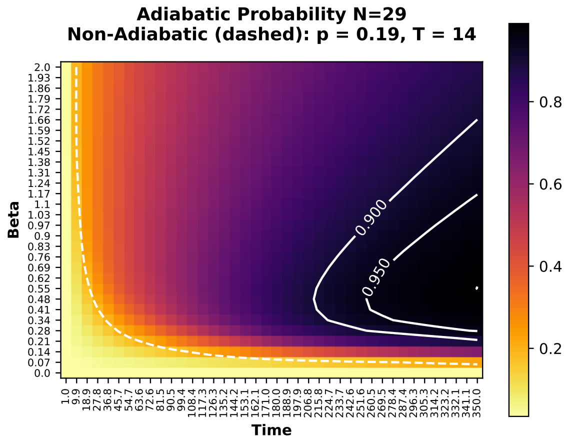
(a) N = 17



(b) N = 21



(a) N = 27



(b) N = 29

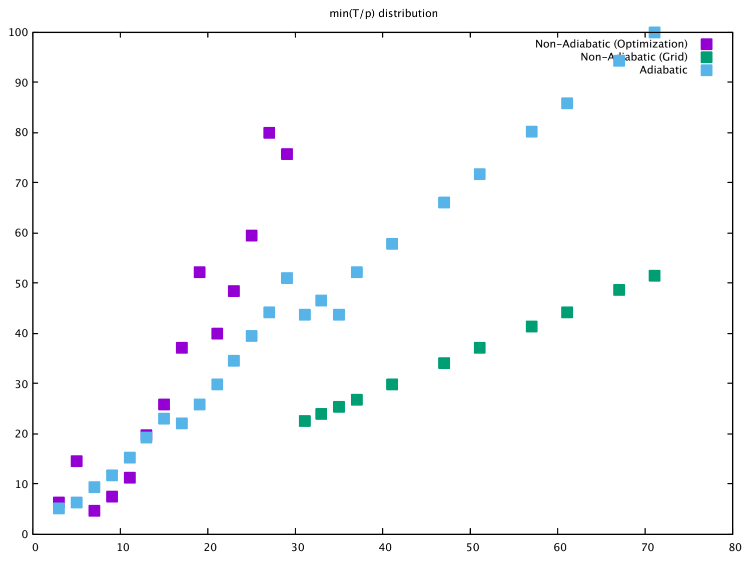


FIG. 4. delta values comparing methods

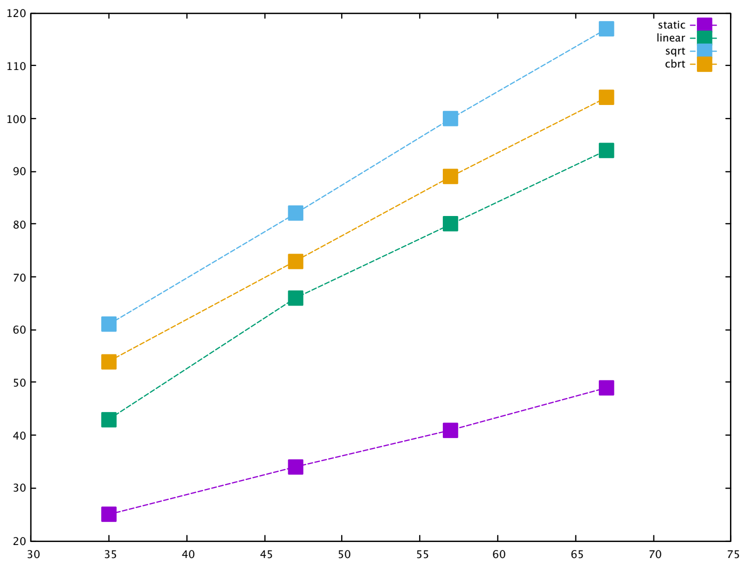


FIG. 5. Delta distrubution for different s(t). Computed for N = 35,47,57,67

As the plot shows however we do not see any speedup for functions that are not linear, on the contrary we do get poorer results. The time-independent implementation remains the best performing one, followed by linear, cubic root and square root.

6. Localization

We can clearly see from the heatmap plots that for large T the probabability gets bigger, and it will eventually get to one. This is, as we already mentioned, a direct consequence of the adiabatic theorem.
A further analysis is due, focusing on studying the probability distribution with a more quantitative approach. That being said the localization is a key feature of the time-dependent approach to quantum search.

C. Complete graph

This section should include a comparison between time dependent and time independent approach to quantum search for the complete graph. Since this topology is solvable analitically I wonder what kind of results should I put in this section.
It might be interesting looking into the order of the probability for p=0.9. Analytically the solution is looked for p=1, possibily (for localization in particular) we could get some speed up over $\sqrt{N}$ for probabilities less than 1. This might work particularly well if the probability is not linear in the vicinity of $\sqrt{N}$
figure: Probability distribution for fixed γ with increasing time

V. FUTURE DEVELOPMENTS

Follows a list of things to be done

- grafo completo

VI. APPENDIX

In this section an overview of the computational methods is presented, focusing the attention on the **optimization algorithm** and the **differential equation solver**. Additionally the normalization error is discussed. Lastly computational reasoning for the **probability grid evaluation** are presented.

Most all numerical simulations were performed using **Python**. Numerical methods such as optimization and ODE Solver come directly from python's native **Scipy**. In addition, a CPU-multiprocessing library, **Ray**, has been used to speed up the grid probability evaluation quite noticeably. Heatmap plots were created using python matplotlib, while additional plots were created with gnuplot.

A. Optimization Algorithm

In Section III a series of benchmark were performed to compare the time-dependent and time independent hamiltonian approach to the search problem. In order to determine which optimization algorithm fitted the best for the task, a number of possible algorithm were tested, such as *shgo, dualannealing, minimize, LHSBH* and *Basin-Hopping*.

Due to the oscillating nature of the probability (a figure is needed) the scipy native **Basinhopping algorithm** was used. As the name suggests the algorithm performs a series of randomized hops, i.e. jumps, of the coordinates in order to find the true maximum This fits particularly well with the series of maxima and minima of the probability function (for fixed γ) in the time-independent hamiltonian.
Additional information on the parameter used are needed (e.g. step size, number of iterations)

B. Schroedinger Solver and Normalization Error

In Section II we presented an evolution which is governed by a time-dependent hamiltonian, used to find the evolved state $|\psi(t)\rangle$. This is accomplished by solving the usual Schroedinger equation using Scipy's **integrate.solve_ivp**, that provides a wide varieties of integrations methods.

As it's routine we used Runge-Kutta RK45, which as stated in the documentation it's a explicit Runge Kutta method of order 4(5). The error is controlled assuming fourth order accuracy, but steps are taken using the fifth-order accurate formula. In addition, the integrator is adaptive, meaning that the time step is chosen for optimal error control. Regarding the error, the algorithm provides two distinct parameters to set a targeted limit, namely the **relative (rtol)** and **absolute tolerances (atol)**. The first provides a relative accuracy, i.e. the number of digits, while the latter is used to keep the local error estimate below the threshold *atol + rtol*abs(y)*. Determining the correct combinations of the two parameters is key for achieving the desired error. A few of those are presented in the following table, where a worst case scenario (N=101, T=1000) is used and the error is evaluated on the expected normalized state. The combination of rtol and atol bolded is the one we used in the solver, which gives us a small enough error on the normalization without greater computational expense.

rtol	atol	norm	error	comp time (s)
e-3	e-5	1.0851	-0.0851	83.8
e-3	e-6	1.0198	-0.0198	19.5
e-3	e-7	1.0188	-0.0188	19.4
e-3	e-8	1.0187	-0.0187	19.5
e-4	e-5	0.9988	1.16e-3	90.6
e-4	e-6	0.9996	3.98e-4	21.7
e-4	e-7	0.9996	4.13e-4	21.4
e-4	e-8	0.9996	4.04e-4	21.5
e-5	e-5	0.9989	1.09e-3	82.8
e-5	e-6	0.9998	1.81e-4	20.5
e-5	e-7	0.9999	1.33e-4	20.6
e-5	e-8	0.9999	1.21e-4	20.8
e-6	e-5	0.9998	2.48e-4	92
e-6	**e-6**	**0.9999**	**5.78e-5**	**22.6**
e-6	e-7	0.9999	2.47e-5	23.2
e-6	e-8	0.9999	1.68e-5	23.2

C. Comments on heatmap and computational time