REPORT:
Quantum search with adiabatic-quantum walk algorithm

Matteo Garbellini
Department of Physics
Università degli Studi di Milano

(Dated: June 24, 2020)

We introduce a novel algorithm for quantum search based on an dynamical implementation of the traditional laplacian hamiltonian, taking advantages of the adiabatic theorem and its implication. We then study the worst case scenario of a simple N-dimensional ring, which gives great localization results but poor time-optimized search results. We then apply the same method to the complete graph, trying to improve the standard continous quantum-walk Grover algorithm. Different time stepping functions are investigated, hopefully giving some useful insight on the dynamic of the constructed hamiltonian.

I. INTRODUCTION

In Section II a review of the topic is presented, i.e. *Continous-Quantum Walk* and *Adiabatic Computation*. Follows a breif overview of Grover Algorithm in its usual Quantum-Walk formulation in Section III, which provides benchmark for the *ring-graph* configuration. Section IV is dedicated to the introduction of the *Adiabatic-Quantum-Walk Algorithm*. Results for two graph configurations (complete and ring) are presented in Section V. The computational methods used in the numerical simulations are discussed in Section VII (or appendix). Section VIII presents a summary of the work done so far, and the further developments to be made.

II. BACKGROUND

A. Countinous - Time Quantum Walk

B. Adiabatic Theorem

A quantum system evolves according to the Schroedinger equation

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle \tag{1}$$

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

$$H(t)|l;t\rangle = E_l(t)|l;t\rangle \tag{2}$$

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

$$\lim_{T\to\infty} |\langle l = 0; t = T|\psi(T)\rangle| = 1 \tag{3}$$

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

$$g_{min} = \min_{0\leq s\leq 1}(E_1(s) - E_0(s)) \tag{4}$$

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

$$T \gg \frac{\varepsilon}{g_{min}^2} \tag{5}$$

where

$$\varepsilon = \max_{0\leq s\leq 1}\left|\left\langle l = 1; s\middle|\frac{dH(s)}{dt}\middle|l = 0; s\right\rangle\right| \tag{6}$$

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

$$H(s) = (1 - s)H_B + sH_P \tag{7}$$

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. SEARCH PROBLEM AND GROVER'S ALGORITHM

IV. ADIABATIC-QUANTUM-WALK ALGORITHM

We introduce an adiabatic-quantum-walk hamiltonian of the form

$$H(t) = \big(1 - g_T(t)\big)L + g_T(t)\beta|w\rangle\langle w| \tag{8}$$

where L is the laplacian described in Section II, $|w\rangle$ is the oracle state, i.e. the target, and $g_T(t)$ is the so-called **step function**, that represents the way (time-wise) the hamiltonian is varied. The idea behind this particular formulation is that we intend to adiabatically *turn on* the oracle, and combined with the adiabatic theorem we should find the solution with high probability.

A. Step Function $g(t)$

The step function $g(t)$ plays a central role in the adiabatic evolution since it regulates how fast the hamiltonian is made to vary, or from the perspective of the oracle, how fast is the oracle switched on. In Section II, we introduced $s(t) = t/T \in [0, 1]$ that regulates the evolution; our $g_T(t)$ goes a little further in the sense that we explore different possible functions to determine which fits the best and, obviosly, produces the highest probability for the lowest time (again, the satisfied adiabatic theorem guarantees convergence to $p = 1$ for $T \to \infty$, so we're not interested in the localization).

In particular we consider the following step functions, all defined in $[0, T]$ with values in $[0, 1]$:

$$g_T(t) = t/T \qquad g_T(t) := \sqrt{t/T} \qquad g_T(t) := (t/T)^{\frac{1}{3}} \tag{9}$$

1. *Comments on the derivative $\frac{d}{dt}H(s)$*

2. *Optimal $g(t)$ for complete graph*

B. Localization vs Optimized Search

We shall describe the possible outcomes of the previously introduced hamiltonian 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. For example, as we shall see later in Section V, the ring-graph is known for not working with traditional quantum-walk algorithm (i.e. found with low probability), but thanks to the adiabatic nature for $T \to \infty$ the solution is found with probability 1.

On the other hand we know that for our algorithm to be somewhat interesting from the perspective of the quantum search we need at least the same speedup we get from the traditional quantum-walk Grover, namely $O(\sqrt{(N)})$. We're thus optimizing for maximum probability and minimum time; 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. It is indeed this what we call *optimized search*.

V. RESULTS FOR SELECTED TOPOLOGIES

A. Ring

B. Complete graph

VI. DISCUSSION

A. So far

B. Future

VII. METHODS

In this section an overview of the computational methods is presented, focusing the attention on specific matters: *Optimization Algorithm* used for determining the optimal γ-configuration for maximum probability, *Schroedinger Solver* for the time dependent hamiltonian evolution and some additional parameters for the optimal *Runge-Kutta Normalization Error*. Lastly computational reasoning for the *2D - Probability Heatmap* are presented.

Most all numerical simulations were performed using **Python** being easy and flexible to sudden changes. 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. 2D - Heatmap plots were created using python matplotlib, while additional plots were created with gnuplot.

A. Optimization Algorithm

In Section II a series of benchmark were performed to compare the standard Quantum-Walk algorithm to the Adiabatic-Quantum-Walk implementation (later simply named 'dynamic'). 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 (cfr figure) 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 (actually, the true *minimum*). This fits particularly well with the series of maxima and minima of the probability function (for fixed γ) in the static algorithm (std QWAlgorithm). Snippets of the parameters used follow.

```
from scipy.optimize import basinhopping
optimization = optimize.basinhopping()
print(optimization)
```

B. Schroedinger Solver

In Section III 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 is used and the error is evaluated on the expected normalized state.

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. 2D - probability heatmap